University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

mgr Grzegorz Góra

Combining instance-based learning and rule-based methods for imbalanced data

PhD dissertation

Supervisor

Professor Andrzej Skowron

(emeritus professor)

December 2021

Author's declaration:
aware of legal responsibility I hereby declare that I have written this dissertation myself and all the contents of the dissertation have been obtained by legal means.

December 30, 2021 .....
date *Grzegorz Góra*

Supervisor's declaration:
the dissertation is ready to be reviewed

December 30, 2021 .....
date *Andrzej Skowron*

Abstract

This thesis presents methods and systems for learning concepts from examples considering two levels of data difficulty represented by balanced and imbalanced data. However, we focus more on imbalanced data.

The RIONA algorithm combines instance- and rule-based approaches. It uses rules with conditions expressing groups of values for both numerical and symbolic attributes. Using the neighbourhood of a test case (instead of the whole training set) of appropriate size, it is both fast and effective (in classification). Additionally, searching for the optimal neighbourhood size is also done efficiently. RIONA is showing the accuracy comparable to the well-known systems.

RIONIDA is an extension of RIONA for imbalanced data. Additionally, RIONIDA combines instance- and rule-based approaches in another aspect, namely by using special rules that are more general than in RIONA. RIONIDA realises a few additional ideas compared to RIONA: optimisation of the explicitly given performance measure, weights for two classes, the idea of scaled rules, optimisation of parameters related to two latter ideas. RIONIDA is relatively fast and significantly outperforms state-of-the-art algorithms analysed in the thesis.

Presented and proved theoretical results for RIONA and RIONIDA help: (i) to make the produced classifiers explainable, and (ii) to accelerate RIONIDA.

Streszczenie

Rozprawa przedstawia metody i systemy uczenia się pojęć z przykładów dla danych zbalansowanych i niezbalansowanych ze szczególnym uwzględnieniem tych drugich.

Algorytm RIONA łączy podejście oparte na instancjach i regułach. Stosuje on reguły z warunkami grupującymi wartości dla atrybutów numerycznych, jak i symbolicznych. Dzięki użyciu otoczenia obiektu testowego (zamiast całego zbioru uczącego) z właściwie dobranym rozmiarem, jest on zarówno szybki, jak i skuteczny (w klasyfikacji). Także wyszukiwanie optymalnego rozmiaru otoczenia jest szybkie. Algorytm RIONA wykazuje skuteczność porównywalną ze znanymi systemami.

Algorytm RIONIDA jest rozszerzeniem algorytmu RIONA dla danych niezbalansowanych. Łączy on podejście oparte na instancjach i regułach w nowy sposób stosując specjalne reguły, bardziej ogólne niż w RIONA. RIONIDA realizuje kilka dodatkowych idei w porównaniu do RIONA: optymalizację jawnie podanej miary jakości, wagi dla dwóch klas, skalowane reguły, a także optymalizację parametrów dla dwóch poprzednich idei. Algorytm RIONIDA jest stosunkowo szybki i daje istotnie lepsze wyniki niż znane algorytmy analizowane w pracy.

Wyniki teoretyczne pomagają w: (i) tworzeniu klasyfikatorów z własnością wyjaśnialności dla obu algorytmów oraz (ii) przyspieszeniu algorytmu RIONIDA.

Keywords

Classification, Supervised learning, Instance-based learning, k nearest neighbours, Rule induction, Multi-strategy learning, Imbalanced learning, Explainability

ACM Computing Classification (rev.2012)

Computing methodologies ↦ Machine learning ↦ Learning paradigms ↦ Supervised learning ↦ **Supervised learning by classification**
Computing methodologies ↦ Machine learning ↦ Machine learning approaches ↦ **Instance-based learning**
Computing methodologies ↦ Machine learning ↦ Machine learning approaches ↦ **Rule learning**
Information systems ↦ Information systems ↦ applications ↦ Data mining ↦ **Nearest-neighbor search**

Tytuł pracy w języku polskim

Łączenie metody bazującej na instancjach z metodą indukcji reguł dla danych niezbalansowanych

Contents

Chapter 1

Introduction

One of the main research domains of Artificial Intelligence (see e.g. [177]) is Machine Learning (ML) (see e.g. [99, 149]). The most common task in ML is classification, which assigns to any given object description a decision from a finite set of decisions.

A specific sub-task of classification is supervised learning [149] (in short, learning). In this sub-task, a finite set of objects (also called cases, examples or instances), labelled by the known decisions, is given. This set is called a training set. The aim is to predict the decision of any new unseen object, called the test object. ML algorithms construct from training sets *classifiers* (usually based on induced data models) that provide decisions for test objects. The thesis distinguishes between *classification algorithm* (in short, *classifier*) and *learning algorithm*. A classifier classifies any test example based on its description, whereas a learning algorithm applies to a wide range of domains producing a classifier based on a given training set. Numerous learning algorithms have been developed so far (see e.g. [99, 149, 216, 226]), yet new ones are still being proposed. The most popular learning algorithms include: decision trees (see e.g. [169]), *rule induction* (see e.g. [72, 146]), support vector machines (see e.g. [203, 235]), *instance-based learning* (see e.g. [7]), simple Bayesian classifiers (see e.g. [54]), artificial neural networks (see e.g. [162]), ensemble learning (see e.g. [50]), and random forests (see e.g. [32]). Within the thesis, we focus on the development of new learning algorithms that draw particularly from two groups of techniques listed above, namely rule induction and instance-based learning.

Recently, much scientific effort has been put into supervised learning that concerns learning from so-called *imbalanced data*. The problem of learning from imbalanced data is well known in the literature (see e.g. [62, 101, 102, 116, 139, 152, 209]). In classification tasks for imbalanced data the correct classification of objects into one specific decision class is much more important than into others. For the classification task with a binary decision, which we focus on in the thesis, there is just one class of special importance. Usually, this class includes a much smaller number of objects than the other one. Therefore it is referred to as the *minority class* and the other one as the *majority class*. As an example of such a problem, one can consider a popular case study from biomedical data analysis related to *Mammography* data set (used in our experiments). It contains images acquired from a series of mammography examinations performed on a set of distinct patients [37, 224]. The objective here is to predict for a new patient, based on the training set of images, whether this

patient is cancerous or healthy. The class of cancerous patients is much smaller and simultaneously much more crucial with respect to correct classification than the class of non-cancerous ones.

At the beginning, while working with imbalanced data, it is worth asking why the standard classifiers (i.e. classifiers induced by learning algorithms designed for balanced data) do not work well with such data? There are at least the following four reasons for that:

- Standard classifiers aim to maximise the classification *accuracy* (expressed by the number of correct predictions made by classifier divided by the total number of predictions made). However, for imbalanced data, this *performance measure* is inadequate.

- The construction of standard classifiers in case of imbalanced data leads to achieving a rather low accuracy rate for the minority class while achieving high accuracy rate for the majority class (see e.g. [102]).

- Standard algorithms identifying noisy examples, i.e. training objects with incorrect decision labels, do not distinguish between the decisions labelling them into majority or minority classes. If an example truly belonging to the minority class is identified as noisy, or a truly noisy example from the majority class is not identified as such, then classification of the objects from the minority class gets complicated (see e.g. [139]).

- Standard classifiers assume equal misclassification costs for all classes. However, the misclassification cost can be often much higher for the minority class than for the majority class (as in the case of the mentioned *Mammography* data set).

In recent years, the problem of learning from imbalanced data (the *imbalanced learning problem*) has become a big challenge (see [229]). Many methods for dealing with this problem have been proposed (see e.g. [31, 36, 38, 62, 95, 101, 102, 116, 121, 139, 152, 195]). Basically, these methods can be divided into two groups: data-level solutions and algorithmic-level solutions (see e.g. [195]).

Data-level solutions transform (using methods called filters) the original data set into a new one and then apply a standard classifier to it. In this approach, one can distinguish the following approaches of data transformation: over-sampling methods, which increase the cardinality of the minority class (see e.g. [20, 37, 63, 179, 204]), under-sampling methods, which decrease the cardinality of the majority class (see e.g. [196, 214]), and hybrid methods, which combine the previous two approaches (see e.g. [14]).

Algorithmic-level solutions relate to the development of algorithms that take into account the problem of imbalanced data. Here, one can distinguish the following approaches: adapting existing algorithms originally developed for balanced data by introducing bias toward the minority class (see e.g. [40, 52, 56, 90, 105, 123, 130, 135, 136, 153, 227, 228]), one class learning (see e.g. [111, 142, 143, 174]), cost-sensitive learning (see e.g. [45, 57, 69, 132]), and ensemble methods (see e.g. [139]).

1.1 Motivations

In the thesis, we focus on a specific approach for imbalanced data combining instance-based learning and rule-based methods. In the past, there have been some attempts to combine instance- and rule-based approaches, however only for balanced data (see e.g. [53, 128]). Nonetheless, at least two reasons are advocating for developing such approaches not only for balanced but also for imbalanced data.

First, both approaches use reasoning schemes easily understandable by a human. Such schemes include rules in the form of

If *some conditions are satisfied,* ***then*** *the decision is X*

which are often used by humans. Analogously, the reasoning scheme of the form

Since *our new example A*
is the most similar to *another known, examined example B,*
then *example A should have the same decision as example B*

used in instance-based learning, is also easily understandable by humans. Because of this, such approaches meet the requirements for ML systems to be explainable. Together with the decision for the given test object the classifying system should provide an explanation for this decision understandable by the user. In the last years, one can observe rapidly increasing importance of this issue in real-life applications. This is related to the topic of so-called Explainable Artificial Intelligence (see e.g. [3, 55, 94]).

Second, there are some intuitions, following from mathematical considerations, suggesting the use of instance-based learning, perhaps in combination with rule induction. The rule-based approaches are examples of a two-stage procedure. At the first stage, we induce (estimate) the unknown decision function. At the second stage, we apply this induced function to classify test examples. However, Vapnik observed that the decision function estimation is a much more general problem than we usually need to solve in practice. In most of the cases, we only need to estimate the unknown decision function at 'a few' new points defined by test objects (see [203, p. 12]). He suggests that if one needs to infer decisions for new cases based on small training sets, one should take into account the following principle:

> *If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem.* [203]

This principle suggests that using instance-based approaches can be relevant. It also applies to methods combining instance-based approaches with other ones.

Well known among instance-based methods are kNN algorithms. This class of algorithms was included in the list of the top ten most influential data mining algorithms [226]. In the simplest case, it returns the decision of the training example

most similar to the test case. In general, these algorithms base the classification on the number of occurrences of classes of the k most similar examples to the test one – forming a set of examples sometimes called the neighbourhood of the test case. The *similarity* is measured by a certain *distance* function, called also metric. The performance of the kNN method strongly depends on the *metric* (distance) used. Numerous papers propose different solutions for inducing a metric from data (see e.g. [106, 127, 186, 230]). Also, the quality of kNN usually strongly depends on the value of k. In practice, estimation of the optimal k is often done by cross-validation technique (see e.g. [180]). Generally, there exist a number of approaches to automatically select the optimal value of k (see e.g. [35, 39, 77–79, 237, 239]).

Rule-based methods represent patterns-laws by *if-then decision rules* relating some conditions with some decisions. Among the rule-based methods, several approaches can be distinguished (see e.g. [72] for an overview of these methods). Generally, rule-based methods can be characterised by their three important components related to the following questions:

1. What is the description language of the rules?
2. How is the set of rules generated?
3. How the obtained set of rules is used for the process of classification? (Usually, it is related to the so-called conflict resolution.)

Considering the first question, the vast majority of approaches use the conjunction of conditions (descriptors) of the form $attribute = value$ in the predecessor of the rule, and $decision = value$ in the successor of the rule. However, other approaches exist as well, e.g. monotonic rules (see e.g. [26]). In relation to the second question, two main approaches can be distinguished: induction of a minimal set of rules (see e.g. [42, 68, 71, 86, 147, 188]), and induction of a non-minimal set of rules (see e.g. [72, 86, 183]). As regards the third question, one can distinguish the following approaches: algorithms producing an ordered set of rules (see e.g. [41, 42]), and different strategies for the conflict resolution (see e.g. [189] for review of this issue).

Among algorithmic-level solutions for imbalanced data, many rule-based methods exist (see e.g. [176], see [152, Chapter 4] for an overview). As mentioned previously, the thesis focuses on methods for the classification of imbalanced data combining instance- and rule-based approaches.

1.2 Aim of the thesis and sketch of the results

The main aim of the thesis is to develop learning algorithms based on the combination of instance- and rule-based methods with the high quality of classification for different types of data sets. We deal with this aim in two steps by proposing the RIONA algorithm for balanced data and the RIONIDA algorithm for imbalanced data. We show on a specific example of the RIONA algorithm how to generalise the algorithm structure for balanced data to make it effective for imbalanced data, which leads to the RIONIDA algorithm. A straightforward approach is to apply a *filter* for imbalanced data before using the RIONA algorithm. However, in the thesis, we

propose a different approach, namely the approach based on a modification of the RIONA algorithm to make it relevant for imbalanced data.

1.2.1 RIONA – an algorithm for balanced data

In the first step we propose the Rule Induction with Optimal Neighbourhood Algorithm (RIONA) [81]. The algorithm combines instance- and rule-based approaches and was developed to be competitive with other methods concerning the performance measure known as the accuracy (see e.g. [109]). The algorithm is based on a few ideas.

- RIONA computes rules in a lazy manner (see e.g. [6]), that is it induces a very limited set of decision rules relevant only for the test example. This is a different strategy than inducing a large number of decision rules in advance to use them during testing.
- The classification performed by RIONA on a given test object is based on rules induced only from the neighbourhood of the given test example. Note that a small number of rules is sufficient when the lazy approach is applied.
- We use a different kind of rules than those commonly used in rule-based approaches, where conditions are of the form: *attribute equal to the specific value*. In RIONA, more general rules are used with conditions of the form: *attribute belongs to a set of values*. These sets of values are specified by grouping both numerical and symbolic values of attributes. In voting for the decision by rules *covering* the example being classified, the aggregation of the support sets of such rules is used.
- RIONA constructs object neighbourhoods of the optimal size.
- The notion of similarity between objects is essential for RIONA for two purposes: (i) constructing the neighbourhood for a given object, and (ii) grouping values of attributes.

The performed experiments reported by the author of the thesis (see [81]) and in the literature (see e.g. [8, 48, 84, 85, 178]) show that RIONA is competitive with many other well-known systems.

1.2.2 RIONIDA – an algorithm for imbalanced data

It turns out that RIONA has some drawbacks characteristic for the standard algorithms running on imbalanced data. Here, comes the second step of the thesis's aim. Now, the objective is to modify the proposed algorithm combining instance- and rule-based methods (RIONA) for improving its performance on imbalanced data. Namely, in the second step we propose the Rule Induction with Optimal Neighbourhood for Imbalanced Data Algorithm (RIONIDA). All the ideas listed in previous subsection for RIONA are also realised in the RIONIDA algorithm. This new algorithm realises a few new ideas in comparison with RIONA.

- RIONIDA tries to maximise not accuracy, but one of the performance measures much more relevant for imbalanced data, like *F-measure* or *G-mean* (see e.g. [18, 109]).

- Conflict resolution of rules in RIONIDA is more sophisticated than in RIONA. The aggregation of decisions of rules covering classified objects is defined using the property that the minority class is 'more important' than the majority class. The phrase 'more important' is expressed by the importance degree. The importance degree of the minority class (and in consequence of the majority one) is tuned during learning.

- Rules *inconsistent* to a certain degree are allowed. The level of inconsistency is also tuned during learning.

RIONIDA significantly outperforms tested in the thesis state-of-the-art methods developed for imbalanced data. This fact is illustrated in the thesis on several benchmarks (see Chapter 5).

The approach used in developing RIONIDA is different from the ones presented in the literature. To our knowledge the only algorithm designed for imbalanced data analysis that combines the instance- and rule-based approaches and at the same time belongs to the algorithmic level approach (which modify algorithms for balanced data) is BRACID (see e.g. [152, 153]). BRACID is a modification of the RISE algorithm to make it applicable for imbalanced data. There are some substantial differences between BRACID and RIONIDA. First, BRACID calculates rules in the learning phase (in advance), while RIONIDA does it in the testing phase (i.e. according to the lazy approach). Second, BRACID starts from rules equivalent to instances and induces quasi-optimal rules for the given data set. RIONIDA adopts a different strategy and takes into account a large space of parametrised rules formulated in a specific language. Note that different parametrisations correspond to different approaches, including a pure instance-based approach, a pure rule-based approach, and the approaches that combine them. For the given data set, RIONIDA selects the optimal parameter settings of rules, and does it very efficiently. Third, BRACID optimises rules for F-measure, while RIONIDA can optimise any performance measure specified by a user (defined on the basis of *confusion matrix*), and does it effectively.

1.3 Comments on some problems related to imbalanced data

In the thesis, we concentrate on some problems concerning the analysis of imbalanced data. However, in general, such analysis may encompass many other issues. Several of them are covered in the thesis only marginally or not covered at all and are left for further studies. Some of them are shortly discussed below.

The solutions presented in the thesis for imbalanced data are directly applicable only for binary classification problems. In the case of imbalanced data sets with multiple-classes, one may solve the original classification problem by transforming it

into a family of binary classification subproblems and appropriately joining the partial solutions (see e.g. [60, 62]). Nevertheless, the classification of imbalanced data sets with a multiple-class problem has its own specific difficulties. Generally, we do not deal with this problem in the thesis.

Closely related to the field of imbalanced data is the problem of *outlier* detection (see e.g. [4, 100]). Moreover, for certain applications, these issues overlap substantially (see e.g. [4]). It is not our aim in the thesis to study the outlier detection problem. Here we would only like to note the relationship between these issues.

In the developed in the thesis algorithms, we also implemented a heuristic for dealing with missing values in data sets. However, the problem of missing values is not the one on which this thesis is focused on. Problems related to the analysis of data with missing values have been studied for many years (see e.g. [88, 182]). This problem was also studied in the context of imbalanced data (see e.g. [234]).

Learning of similarity measure between objects (or metric) is the crucial issue for instance-based approaches. In the thesis, we use metrics[1] that are well known from the literature. These metrics depend on data sets (are computed using the training set). The problem of learning metrics (or similarity) from data is a separate issue widely studied in the literature (see e.g. [19, 107]).

Our approach encompasses the grouping of attribute values. This can be considered as searching for new features. However, in the thesis, we do not deal in a deeper sense with the problem of searching for new features as, for example, deep-learning does (see e.g. [105]). In a sense, we try to explore what can be done by using either basic or compound, but not very sophisticated, features which are 'close to features defined directly on sensory data'.

The algorithms presented in the thesis could be used to construct more compound classifiers, e.g. ensemble classifiers (see e.g. [139]) with use of RIONIDA (or/and RIONA) and other state-of-the-art algorithms. Such combinations can potentially produce classifiers with better performance than each of the partial classifiers. We treat it as a separate investigation issue.

It should be noted that solving real-life problems with imbalanced data often may require developing new methods, different than the ones used in the thesis. In general, one should be able to join such models as presented in the thesis and logical reasoning using expert-knowledge, e.g. expressed in a fragment of natural language (see e.g. [200], [232], [165]). One must also take into account the fact that the reasoning often should concern not only a static world but also one that changes over time (see e.g. [199, 207]). Moreover, in practice, it is often essential to analyse what will happen if we take action related to the decision indicated by the classifier. In other words, one should assess the risk by applying risk management techniques, which become more and more important in recent years (see e.g. [104], [119]).

In the thesis, we use *a priori* fixed performance measure. In practice, it may need to be learned based on data from an expert, dialogue with an expert, or background knowledge. Moreover, the perception processes grounded in the real world, related to classified complex situations should be taken into account. The mentioned above issues are essential for real-life problems, although not covered in the thesis.

[1] In fact we use metrics or pseudometrics. For simplicity we do not distinguish between them in this introductory chapter.

1.4 Results of the thesis

The main results of the thesis consist of construction and analysis of two learning algorithms: RIONA and RIONIDA. The first algorithm is dedicated to balanced data, while the second one is dedicated to imbalanced data. The RIONA algorithm and its primary analysis were done jointly by the author of the thesis and Arkadiusz Wojna (see [81–83]). The work concerned with the RIONIDA algorithm was done by the author of the thesis.

In the thesis, we focus more on the imbalanced learning problem. Therefore the RIONIDA algorithm is the most crucial for the thesis. Nevertheless, the RIONA algorithm is an essential step in constructing RIONIDA. However, as RIONA is relevant only for balanced data, we do not present a full analysis of it. In particular, we do not present in detail the comparison of RIONA with other algorithms relevant for balanced data that are known from the literature, but we only add references to the published papers related to the RIONA performance where such comparisons are included.

The main idea of the RIONA algorithm is to combine the two widely-used empirical approaches to learning from examples, namely instance-based learning and rule induction. The RIONA algorithm possesses several properties important for constructing appropriate classifiers for balanced data. Constructing an algorithm that provides all these properties is a challenge and constitutes a significant result of the thesis. Below, we shortly describe these properties.

1. The RIA algorithm is a particular case of RIONA with the whole support set (i.e. the whole training set is treated as the neighbourhood of the test case). RIA implements the previously quoted idea of Vapnik: 'try to solve the problem directly and never solve a more general problem as an intermediate step' and has a very interesting and practical property. Namely, RIA is equivalent (relative to classification) to the algorithm which, in an intermediate step, generates all *consistent* and *maximally general* rules. The latter algorithm has exponential time complexity, while RIA has much lower – quadratic one. Moreover, in particular, the RIA algorithm (and RIONA) does not require discretisation (or value grouping). It adequately groups values for both numerical and nominal attributes while generating rules.

2. In the general case of RIONA, the decision is predicted based on a support set restricted to a neighbourhood of the test case rather than the whole support set of all rules covering the test case[2].

3. The size of the optimal neighbourhood is automatically induced during the learning phase. It is worthwhile to mention that the learning of the optimal neighbourhood is based on the idea of dynamic programming (see e.g. [43]), which makes the computational time complexity of this step low. Moreover, the empirical study showed an interesting fact that it is enough to consider a small neighbourhood to achieve classification accuracy comparable to the algorithm

[2]It should be noted that a specific metric for symbolic attributes, known as SVDM in the literature (see e.g. [53]), is used for finding objects similar to a given test object.

induced from the whole learning set (see e.g. [184] for the algorithm computing the complete set of consistent and maximally general decision rules). Thus, the combination of kNN and a rule-based algorithm leads to a significant speed-up of both learning and testing phase in comparison with the RIA algorithm using all maximally general rules.

4. The method is competitive with other approaches known from the literature [7, 128, 169] from the perspective of predictive quality. In particular, the presented classifier has a high accuracy for two kinds of data sets: the more suitable for kNN classifiers and the more suitable for rule-based classifiers.

5. The theoretical results formulated and proved in the thesis show the relationships of the RIONA classifiers to both instance- and rule-based classifiers. In particular, we show the equivalence (relative to the classification) of the RIONA algorithm with the rule-based algorithm generating all consistent and maximally general rules from the neighbourhood of the test case. Consequently, the RIONA classifier can be represented by a rule-based classifier, with rules easily interpretable by humans. These theoretical results provide the explainability of the resulting classifiers of RIONA and could be used in the situation when an explanation or justification of the derived decision is important.

Moreover, we proposed the Optimal Nearest Neighbour algorithm (ONN), which is a simple modification of the RIONA algorithm. In ONN, instead of using rules, the kNN method is used for the constructed neighbourhood. ONN uses the same metric as the RIONA algorithm and learns the optimal neighbourhood in a similar way. There are two reasons for mentioning this algorithm here: (i) for some data sets this algorithm has better performance than RIONA, and (ii) this fact is used in the construction of the RIONIDA algorithm (see the forthcoming discussion on RIONIDA).

However, the RIONA algorithm is not suitable for imbalanced data, due to the reasons listed previously (on page 10). Below, we refer to them explaining why RIONA does not perform well for such data.

- RIONA tries to maximise accuracy. This measure assigns equal misclassification costs to the minority class and the majority class. However, this approach is not suitable for imbalanced data.

- RIONA implicitly assumes balanced class distribution. This means that it does not properly deal with data such that for many objects from the minority class their neighbourhood contains overwhelmingly many objects from the majority class. Then, there are also more objects from the majority class supporting rules constructed for objects from the minority class. In consequence, many test examples from the minority class may be misclassified as belonging to the majority class.

- One may obtain a high accuracy rate with low accuracy for the minority class. This fact causes that the RIONA classifier is not acceptable for imbalanced data classification.

The RIONIDA algorithm is based on a modification of the RIONA algorithm. It aims to develop classifiers for imbalanced data with the highest possible quality. To make the task simpler, in RIONIDA, the number of decision classes is limited to two only, i.e. RIONIDA is directly applicable only for binary classification problems. The RIONIDA algorithm, analogously to RIONA, is based on a combination of instance-based learning and rule induction. However, while constructing RIONIDA, some substantial changes have been introduced compared to RIONA. These changes allowed us to obtain an algorithm, which is a significant result of the thesis. This algorithm has the following important properties.

1. RIONIDA performs optimisation during the learning phase not relative to accuracy, but relative to a measure more relevant for imbalanced data (e.g. F-measure or G-mean).

2. Because for the imbalanced learning problem the correct classification to the minority class is more important than to the majority class, the minority class is treated in a special way during the conflict resolution (i.e. method of choosing the final decision if there is some evidence for both the minority class and the majority class). Another problem is related to the choice to what extent the minority class is more important than the majority class.

3. As the ONN algorithm for some data sets gives better results than the RIONA algorithm, we decided to combine the strengths of both of them. RIONIDA can use the rule-based approach, the instance-based approach or a combination of these two. This selection is realised using a parameter representing the degree to which using rules in the neighbourhoods is considered to be relevant[3].

4. All the main (internal) parameters of the RIONIDA algorithm are automatically induced during the learning phase. Let us recall that these parameters consist of the neighbourhood size (this feature is adapted from RIONA), the importance degree of the minority class, and the allowed level of inconsistency. The last one specifies to what extent the rule-based approach (or inversely the instance-based approach) is used. Again, it is important to stress that we present efficient in time methods for learning all these parameters using the dynamic programming technique. Moreover, we introduced the possibility to further accelerate RIONIDA and to reduce its space complexity.

5. For G-mean and F-measure, two theorems provide estimates of the optimal degree of importance of the minority class under the assumption of a 'totally random' distribution. These estimates are faster alternatives than solutions given by parameter learning, and can be used for setting the default value for the appropriate parameter in RIONIDA. Moreover, an interesting conclusion follows from these theorems. Namely, for a certain class of classifiers the optimal

[3]Let us note that we still use instance-based approach to build neighbourhood as it was mentioned previously in discussion on RIONA. Thus the RIONIDA algorithm combines instance- and rule-based approaches in two aspects. First, it uses instance-based approach to limit the neighbourhood that we take into account (e.g. for rule generations). Second, it uses rule-based approach or instance-based approach or even approach 'between' these two.

one might be significantly different (relative to classification) for different performance measures. Additionally, the assessments of such two optimal classifiers may be significantly different depending on the performance measure used for the assessment. The practical implication for real-life classifications is that without a precise specification of the particular performance measure we are interested in, the 'best classifier' term can be ambiguous or even misleading.

6. RIONIDA performs significantly better than tested in the thesis state-of-the-art algorithms known in the literature. We performed the comparison of RIONIDA quality with all main well-established algorithms whose codes were available to the author of the thesis [7, 37, 42, 53, 68, 90, 153, 169, 190, 214]. Such a choice guaranteed the reproducibility of experiments. The superiority of RIONIDA was demonstrated in experiments on benchmarks, using performance measures relevant for imbalanced data. The comparison tests were thoroughly designed using the current knowledge on the evaluation of learning algorithms in the context of imbalanced data. In particular, we took into account the appropriate performance measures, their proper estimation method, which is a complex problem by itself, proper data sets selection, and finally the possibility of different algorithms settings. A statistical evaluation of the obtained results is also included. Let us mention that RIONIDA performs significantly better than RIONA boosted by relevant filters (RIONA with a data-level approach).

7. RIONIDA has the desired property of explainability, which is mainly provided by theoretical features of RIONA that are described in point 5 of RIONA properties.

8. Most of the above-listed features of RIONA also apply to RIONIDA. In particular, RIONIDA, analogously to RIONA, does not require prior discretisation or value grouping. Moreover, for certain settings, RIONIDA is equivalent to RIONA.

To sum up, the RIONA algorithm is a learning method which is efficient in time with a good performance for balanced data. The RIONIDA algorithm is the combination of RIONA and ONN algorithms (and their further extension), designed for dealing with imbalanced learning problems (although limited to binary classification). Importantly, RIONIDA performs significantly better than the state-of-the-art algorithms designed for dealing with imbalanced data and at the same time has a relatively low computational complexity. In particular, RIONIDA significantly outperforms RIONA with filters, which is the common, straightforward adaptation of a standard algorithm for imbalanced data.

Finally, we would like to mention two minor results of the thesis. The first one is the proposed methodology approach with three levels of comparison of learning algorithms taking into account many variants of algorithms, including their non-default parameter settings (see the discussion about experiments in Chapter 5). The second one is the construction of the example leading to different conclusions on which algorithm is better depending on the method of aggregation of partial cross-validation results. More precisely, in the case of macro-averaging, one

algorithm outperforms the other, while for micro-averaging the other way round (see Appendix B).

Last but not least, the above-mentioned results concerning RIONA and RIONIDA might be seen as examples of a few abstract and general directions of research for effective and efficient learning algorithms. We hope that some of our ideas may be adapted in projects in which the design of learning algorithms is based on concepts other than those used in the thesis. First, we showed that in the case of rule-based classifiers computation of the measure for conflict resolution based on all consistent and maximally general rules can be significantly accelerated by using the lazy approach – a similar approach might be used for other measures for conflict resolutions. Second, we showed that combining instance-based learning with another method such as the rule-based approach can be beneficial both in terms of quality and efficiency – that path might also work for approaches different from the rule-based approach, e.g. decision trees. Third, we showed that parametrisation of classifiers based on the lazy-based approach can be realised much more effectively with the use of dynamic programming than by direct computation – that approach might be applied for other algorithm architectures and/or other parametrisations. Fourth, we showed an example of how a learning algorithm for balanced data can be successfully modified into a learning algorithm for imbalanced data – an analogous modifications could be realised for other algorithms dedicated to balanced data.

1.5 The organisation of the thesis

The thesis is divided into six chapters. The introductory chapter, in particular, explains why it is important to develop high-quality classifiers for imbalanced data and describes the main results of the thesis. Chapter 2 presents the basic concepts and introduces notation used in the subsequent chapters. In particular, the main approaches and examples of specific methods for imbalanced learning problem are outlined. Chapter 3 describes the RIONA algorithm designed for balanced data. RIONA can be treated as the basis for the RIONIDA algorithm. In particular, the theoretical results concerning the RIONA algorithm are included. Chapter 4 introduces the RIONIDA algorithm, a modification of RIONA, designed for classification of imbalanced data. Chapter 5 describes the results of experiments in which the proposed algorithm was compared with some state-of-the-art algorithms for imbalanced data on benchmarks and real-life data sets. Finally, the concluding remarks are placed in Chapter 6. The three short appendices (A, B, and C) complement Chapters 3, 2, and 4, respectively, by adding some details, which left in the chapters could break the main flow of thought. At the end of the thesis are included: the index, lists of abbreviations and symbols.

1.6 Collaboration

Most of the results presented in Chapter 3, especially the development of the RIONA algorithm, were carried out in collaboration with Wojna. The common results are published in the papers [81–83]. In the mentioned cooperation, the contribution of

both authors was roughly equal. Additionally, both authors independently extended the common research in different directions.

The results of the work of Wojna are published in his PhD thesis (see [219]) and other papers (see [185, 217, 218, 220]). Let us only mention those results which the author of the thesis used in its experimental part. Wojna expanded the developed RIONA algorithm in two directions. First, to make it possible to work with many other possible metrics and weighting attribute methods. Second, the research on an acceleration of the algorithm was carried out.

Independently, the author of the thesis developed a new form of presentation of foundations leading to RIONA. It enabled him to make the presentation of RIONA in a more transparent way. Moreover, some facts included in the thesis (Theorem 3.11, Corollary 3.12) better explain the relationships of RIONA with rule-based classifiers. On the basis of these theoretical results, the method of explaining the resulting classifier of RIONA for a human is proposed.

Moreover, the author of the thesis extended the RIONA algorithm to work with imbalanced data. As a result of this research, the RIONIDA algorithm was developed. Chapter 4 presents this algorithm, and Chapter 5 presents the experimental comparison of the developed algorithm with some other methods developed for imbalanced data. Experiments for RIONIDA use acceleration of the RIONA algorithm. Additionally, the developed metrics and weighting methods developed for the RIONA algorithm described in [219] were also tested.

For comparison, we used methods from two sources:

1. available in WEKA (see e.g. [2, 215]) and

2. methods provided by members of the prof. Jerzy Stefanowski team.

Also, some scripts for testing different methods were provided by the team of Stefanowski.

1.7 Software

The software for the RIONA algorithm is publicly available for use as part of the open-source Java library available at `http://rseslib.mimuw.edu.pl` (see [1, 221, 222]). RIONA can be used within WEKA after installing Rseslib package in Weka Package Manager. Information on how to run RIONA both in WEKA and natively can be found in Rseslib User Guide available at the library web page (see [222]).

The RIONIDA algorithm is planned to be publicly available analogously to RIONA as a part of the open-source Java library. Especially RIONIDA is planned to be used within WEKA as a part of the Rseslib package in Weka Package Manager. Currently, it is available for the use of reviewers on request[4].

[4] ggora@mimuw.edu.pl

Acknowledgements

First and foremost, I would like to express my immense gratitude to my supervisor, prof. Andrzej Skowron for his guidance, patience, and invaluable support both on the academic and personal plane. His example has been inspiring me for all those years.

Secondly, I wish to thank Arkadiusz Wojna for past collaboration as well as for his insightful suggestions and always being ready to help in this project. I would also like to express my heartfelt gratitude to prof. Jan Bazan for his precious feedback, assistance, and our previous collaboration on a very practical problem of planning of the treatment of infants with respiratory failure. Although this research is not mentioned in this thesis at all, it has led me to work on the problem of imbalanced data at a more general level. I am also very grateful to prof. Jerzy Stefanowski and his team for their support and advice; in particular, to dr Krystyna Napierała for sharing her BRACID software and to prof. Szymon Wilk for technical consultations.

I would also like to acknowledge: Soma Dutta, Adam Sikora, Bartosz Pióro, Jarosław Pióro, and Piotr Buczkowski for linguistic consultations and corrections to the draft; Zuzanna Szymańska and Liliana Trzpil for their help in revising the bibliography; prof. Janez Demšar and dr. George Forman for consultations related to experiments, prof. Błażej Miasojedow for statistics consultations, prof. Stan Matwin for bibliographic suggestions, and prof. Nitesh Chawla for providing the *mammography* data set; members of the (former) Group of Mathematical Logic, including Andrzej Janusz, Wojciech Świeboda, Paweł Gora, Marcin Szczuka, Sinh Hoa Nguyen, Hung Son Nguyen, Andrzej Jankowski, Dominik Ślęzak, Piotr Wasilewski, Marek Grzegorowski, and others for assisting me throughout the project. Last but not least, I would like to thank my family, friends, and the brothers and sisters of my order who in different ways have supported me in my work.

✝ J.M.J.

Chapter 2

Basic notions

This chapter presents the fundamental concepts used, in particular those defined for the purpose ot the thesis.

The following section presents a more formal description of the specific type of learning concepts from examples. Section 2.2 introduces an important notion for instance-based learning, namely metric (and *pseudometric*). In particular, pseudometric used in the thesis is defined. Section 2.3 discusses more formally two essential for the thesis methods in machine learning, i.e. rule-based methods and instance-based learning. Also, essential for the thesis, the lazy rule learning approach is introduced. Section 2.4 discusses the difficulties of learning from imbalanced data and outlines the currently existing methods. In Section 2.6, we enumerate a few important steps of evaluation of learning algorithms dealing with imbalanced data.

2.1 Learning concepts from examples

In this section, we present a more formal description of supervised learning. In supervised learning, it is assumed that the training examples are classified (labelled) by class labels. The goal is to learn a decision function that maps inputs defined by a vector of values of attributes on objects to outputs representing the values of decision function (decision attribute) using training examples described by inputs and their desired outputs.

The domain of learning is a space of objects $\mathbf{X}$. Each object $x \in \mathbf{X}$ is described by a finite set of pairs $(a, a(x))$, where a is a conditional attribute from a given set A of (conditional) attributes, i.e. $a : \mathbf{X} \to V_a$ for $a \in A$, where the codomain V_a of a is the set of values of a and $a(x)$ is the value of a on the object $x \in \mathbf{X}$. We consider two types of attributes: numerical and symbolic[1]. We denote the sets of symbolic and numerical attributes respectively by A_{sym} and A_{num}.

The values of numerical attributes are comparable and can be represented as (real) numbers ($V_a \subseteq \mathbb{R}$). In practice, these are integer or real values. Without loss of generality, we assume that V_a is equal to an interval (l_a, u_a), where $l_a, u_a \in \mathbb{R}$ (possibly not all of the values from the interval are used).

[1] In fact, we consider also the third type of attributes, ordinal attributes, which are attributes with a linear order. However, such attributes are represented as integer numbers and are regarded as sub-case of numerical attributes.

Symbolic attributes have incomparable values (e.g. related to colour, shape, language). The codomain of a symbolic (discrete-valued) attribute is a finite set, i.e. $V_a = \{v_1, \ldots, v_l\}$ for some $l \in \mathbb{N}$.

Let us present an important notion of concept over $\mathbf{X}$. A concept is any subset of $\mathbf{X}$. Thus, we can represent it by the characteristic function, i.e. a binary function $c : \mathbf{X} \to \{positive, negative\}$ with codomain containing two values: *positive* representing the members of the concept and *negative* representing the non-members of the concept.

In general, one can represent many concepts simultaneously by a decision function $d : \mathbf{X} \to V_d$, where V_d is a finite set of decisions. We assume that $d \notin A$. Let n_d denote the cardinality of decision value set V_d, i.e. $n_d = |V_d|$ (n_d is the number of decisions). Thus we can write $V_d = \{d_1, d_2, \ldots, d_{n_d}\}$, where $1, 2, \ldots, n_d$ are indexes of the decision classes. Without loss of generality we also assume that the decision value set is represented by a set of consequent natural numbers, i.e. $V_d = \{0, 1, \ldots, n_d - 1\}$. In the thesis, we generally, do not use any relation between decision classes expressed in terms of their indexes.

Each value d_i in the set V_d characterises a separate concept (decision class) $\{x \in \mathbf{X} : d(x) = d_i\}$, denoted by $Class(d_i)$. In the thesis, we consider many-valued decision function ($n_d \geq 2$) for balanced data and binary-valued decision function ($n_d = 2$) for imbalanced data. In the latter case, we use notation $V_d = \{d_{maj}, d_{min}\}$, where d_{min} indicates the *positive*, *minority class* of our main interest and d_{maj} indicates the *negative*, *majority class* of less importance.

The goal of learning is to approximate the target decision function on the whole domain $\mathbf{X}$ on the basis of the provided finite number of examples. This set of provided examples is called the training set and is denoted in the thesis by *trnSet*. Each training example is described by values of all attributes and value of the decision function. We can represent the training set by a data table. Rows in the data table correspond to objects (training examples) to be analysed in terms of their properties (attributes) and the concept (class) to which they belong. In Table 2.1, an exemplary table representing a training set and one test element are presented.

Without loss of generality, we assume that training sets are consistent, i.e. there are no two objects with all conditional attributes equal and with different decision values. This simplifies the notation and proofs. For simplicity, we also assume that there are no missing attribute values for any objects. However, the learning algorithms developed in the thesis also can work with inconsistent training sets which may contain missing values (see, e.g. Subsection 3.3.1).

Definition 2.1. *Any triplet* $(\mathbf{X}, A, d)$*, where* $\mathbf{X}$ *is the space of objects,* A *is a set of attributes and* d *is a decision function, is called* decision system.

Training sets are used to build (induce) *classification algorithms* (in short, *classifiers*). A *classifier* defines a function that, for a given input example, assigns that example to one of n_d classes. A *learning algorithm*[2] computes a function that for a given training set, constructs a classifier (see e.g. [51]).

[2]It should be noted that sometimes in the literature, there is no clear distinction between classifiers and learning algorithms.

Object	*Age*	*Weight*	*Gender*	*BloodGroup* (*BG*)	*Diagnosis*
trn_1	35	90	M	A	Sick
trn_2	40	65	F	AB	Sick
trn_3	45	68	F	AB	Healthy
trn_4	40	70	M	AB	Healthy
trn_5	45	75	M	B	Sick
trn_6	35	70	F	B	Healthy
trn_7	45	70	M	0	Healthy
tst	50	72	F	A	?

Table 2.1: An exemplary data set with 4 conditional attributes (*Age* and *Weight*, numerical; *Gender* and *BloodGroup*, symbolic) and decision attribute *Diagnosis*. Seven objects are from training set and the last object is a test object (its decision is unknown).

The goal of supervised learning in a single application domain is to build a classifier on the basis of a training set, which is equal or close to the target decision function. One can search for such a classifier using the best learning algorithms for this domain. The goal of ML, in general, is to construct learning algorithms which perform well over a wide range of real-life domains and their corresponding training sets (see e.g. [51]). Learning theory provides precise definitions of the concepts used here such as 'close to ...', 'best ...', 'perform well' etc. (see e.g. [10, 149]).

Let us sum up, what was discussed above and additionally indicate the convention of notations used in the thesis. Usually, we assume in the text that a decision system $(\mathbf{X}, A, d)$ is given. Sometimes we assume the decision system to be a *pseudometric decision system* (see Subsection 2.2.3). Whenever we write *object* x or *example* x, we mean that $x \in \mathbf{X}$. Usually, we assume that a training set, $trnSet \subseteq \mathbf{X}$ (normally, $trnSet \subset \mathbf{X}$) is given. Whenever we write *training object trn* or *training example trn*, we mean that $trn \in trnSet$. Whenever we write *test object tst* or *test example tst*, we mean that $tst \in \mathbf{X}$ (however usually $tst \in \mathbf{X} \backslash trnSet$). Throughout the thesis, *trn* and *tst* denote some training example and test example, respectively. Theoretically, for all objects (however, in practice only for test objects) learning algorithm can acquire values for all attributes. For training objects, learning algorithm can additionally acquire values for decision attribute.

2.2 Similarity and metrics in machine learning

In ML, reasoning and learning from cases can be performed using a concept of *similarity*. For example, instance-based learning is based on the assumption that the decision for a new test case can be inferred from the description of the objects similar to the test case. There are many definitions of similarity measures (see e.g. [12, 28, 107, 160]). Generally, objects are considered similar if they have a high degree (fixed *a priori*) of similarity and non-similar if they have a small degree (fixed *a priori*) of similarity. Numerous similarity measures are based on the notions of metrics and pseudometrics (also called *distance functions*). The main idea is that objects close to each other are regarded to have a high degree of similarity, and conversely, objects which are very distanced from each other are regarded to be

non-similar (small value of similarity). In the thesis, we mainly use a weaker concept than metric, i.e. pseudometric (see e.g. [33]).

Definition 2.2. *A function $\varrho : X \times X \to \mathbb{R}$ is a* pseudometric *on the set X (or* distance function *or simply* distance*) if and only if, for all $x, y, z \in X$, the following conditions hold:*

1. $\varrho(x, y) \geq 0$ *(non-negativity)*
2. $\varrho(x, x) = 0$
3. $\varrho(x, y) = \varrho(y, x)$ *(symmetry)*
4. $\varrho(x, z) \leq \varrho(x, y) + \varrho(y, z)$ *(triangle inequality)*

A pair (X, ϱ), where ϱ is pseudometric is called a pseudometric space.

The first condition could be omitted because it is implied by the others. The *metric* additionally satisfies the following condition $\varrho(x, y) = 0 \Rightarrow x = y$. However, we intentionally defined pseudometric as this condition is generally not satisfied for the distance function used in the thesis.

For a given pseudometric ϱ over X, we define *closed ball* (in short, *ball*) of radius $r \geq 0$ centred at $x \in X$, denoted by $B(x, r)$, as the set of all points of X of distance less or equal to r from x, i.e. $B(x, r) = \{y \in X : \varrho(x, y) \leq r\}$. If we use this notation, it is usually clear from the context which pseudometric is used. Otherwise, we write explicitly which pseudometric is used in the closed ball definition. Generally, we use this definition in the case when X is a finite set. Note that if a pseudometric is not a metric, then closed ball with the radius equal to zero may contain more than one element.

The following fact about pseudometric space, which is well known and easy to prove, is important for us[3].

Fact 2.1. *Let $n \in \mathbb{N}$ and, for each $i = 1, 2, \ldots, n$, (X_i, ϱ_i), be a pseudometric space. Then the following product*

$$\prod_{i=1}^{n} X_i = \{(x_1, \ldots, x_n) : x_i \in X_i, i = 1, \ldots, n\}$$

with the function

$$\varrho\left((x_1, \ldots, x_n), (y_1, \ldots, y_n)\right) = \sum_{i=1}^{n} \varrho_i(x_i, y_i)$$

is a pseudometric space.

In the thesis, we use pseudometric which is a sum of pseudometrics defined for each attribute. We assume that for each attribute $a \in A$ a pseudometric $\varrho_a : V_a \times V_a \to \mathbb{R}$ is given. If there are m attributes we define pseudometric on the $\mathbf{X}$ set using

[3] Originally the fact concerns the metric; however, as we mentioned we focus on pseudometrics.

pseudometric on the Cartesian product set $\prod_{i=1}^{m} V_{a_i}$ defined as in Fact 2.1. Finally, the distance between two instances $x, y \in \mathbf{X}$ is defined by:

$$\varrho(x, y) = \sum_{a \in A} \varrho_a(a(x), a(y)) = \sum_{a=1}^{m} \varrho_{a_i}(a_i(x), a_i(y)) \tag{2.1}$$

From Fact 2.1, we have that $(\mathbf{X}, \varrho)$ is a pseudometric space[4].

Now, to define completely a pseudometric, we have only to define pseudometric ϱ_a for each attribute $a \in A$. Usually, this is done separately for numerical ($V_a \subseteq \mathbb{R}$) and symbolic attributes ($V_a = \{v_1, \ldots, v_l\}$ for some $l \in \mathbb{N}$).

2.2.1 Metrics for numerical attributes

The widely known metric on $\mathbb{R}$ is determined by the absolute-value function on $\mathbb{R}$. It is the function $\varrho : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ defined by $\varrho(a, b) = |a - b|$, where $a, b \in \mathbb{R}$. This metric is called the *Euclidean metric* on $\mathbb{R}$.

Considering Equation 2.1 and the fact that the scale of different numerical attributes can be different, we need to make normalisation for each attribute. This is due to the fact that having no prior knowledge we attempt to assign to all attributes the equal importance in measuring the distance (and in consequence similarity). We use normalisation based on the values occurring in the training data set *trnSet*. A commonly used approach for attribute value similarity is to normalise the value difference by its largest observed value difference:

$$\varrho_a(v, w) = \frac{|v - w|}{a^{\max} - a^{\min}} \ , \tag{2.2}$$

where $v, w \in V_a$, $a^{\max}$ and $a^{\min}$ are the maximal and the minimal value for an attribute a among training examples *trnSet* (without loss of generality we assume that $a^{\max} \neq a^{\min}$). There are other possibilities of normalisation, e.g. defined by the standard deviation (see e.g. [219]). However, we use, in the thesis, only the presented one.

For all numerical attributes the final metric ϱ from Equation 2.1 and with no normalisation in ϱ_a is known in the literature as *city-block metric* (also known as *taxicab metric*, *Manhattan metric* or L^1 distance). In the thesis, a *normalised city-block metric* is a metric having components ϱ_a with normalisation, i.e. defined in Equation 2.2.

2.2.2 Metrics and pseudometrics for symbolic attributes

For every set X, there exists a metric. That is the *discrete metric* which is defined on set X by assuming that the distance from each point of X to itself is 0 and distance from each point to every other point of X is 1. We use this metric in the thesis for codomains of symbolic attributes (which are finite sets). For symbolic attribute $a \in A$, we define the *discrete metric* by:

[4] Of course we use also here the following fact. If $f : X_1 \to X_2$ is a function and d_2 is a pseudometric on X_2, then $d_1(x, y) = d_2(f(x), f(y))$ gives a pseudometric on X_1. Let us note that for metric space analogous fact does not hold. This is one of the reasons why we use pseudometrics instead of metrics in the thesis.

$$\varrho_a(v,w) = \begin{cases} 0, & \text{if } v = w, \\ 1, & \text{if } v \neq w. \end{cases}, \text{ where } v, w \in V_a \tag{2.3}$$

For all symbolic attributes with discrete metrics, the final metric ϱ from Equation 2.1 is known in the literature as *Hamming metric*. This metric measures the number of attributes at which the corresponding values are different.

If there is no information about the relations between values of symbolic attributes, this seems the only reasonable metric for this set. However, for a training set of examples, additional information about the value of the decision attribute is given. The decision distribution can be used to compute the distance of two values from codomain of any symbolic attribute. This fact was first used in [187] to define the *Value Difference pseudoMetric* (VDM)[5] with additional weighting of attributes. Later in [53], a simplified version without weighting attributes was used and is known as *Simplified Value Difference pseudoMetric* (SVDM). Thus for symbolic attributes a more informative alternative than Hamming metric is SVDM. For symbolic attribute $a \in A$ we define it by:

$$\varrho_a(v,w) = \sum_{d_j \in V_d} |P(d = d_j \mid a = v) - P(d = d_j \mid a = w)|, \tag{2.4}$$

where $v, w \in V_a$. Originally, it was defined as the sum of absolute values of q powers (for $q = 1, 2, \dots$) of these differences. In the thesis, we use only version with $q = 1$, i.e. defined by Equation 2.4. In practice, the estimation of probability $P(d = d_j \mid a = v)$ is calculated using available training set $trnSet$:

$$P_{trnSet}(d = d_j \mid a = v) = \frac{|\{trn \in trnSet : d(trn) = d_j \wedge a(trn) = v\}|}{|\{trn \in trnSet : a(trn) = v\}|}.$$

SVDM considers two symbolic values similar (i.e. to have small distance) if they have similar decision distribution, i.e. if they correlate similarly with the decision. We may say that this pseudometric is induced from the training set of examples. It strictly depends on the used training set.

It is easy to check that SVDM, in fact, is pseudometric, but not metric. There may exist two different values v, $w \in V_a$ from codomain of a for which $\varrho_a(v,w) = 0$, i.e. on the training set, the distribution of decision can be identical for both groups of objects $trn \in trnSet$ characterised by values $a(trn) = v$ and $a(trn) = w$. In consequence, if in Equation 2.1 at least one component is SVDM pseudometric, then ϱ is pseudometric, but not metric. It is another reason why we use in the thesis mainly the concept of pseudometric.

As an example for SVDM pseudometric let us take into consideration Table 2.1 and symbolic attribute *BloodGroup* (in short, *BG*). We consider only training examples from this table (trn_1, ..., trn_7) and distribution of decision *Diagnosis* is computed over this part of table. Taking this into account we obtain the following distances

[5] Strictly speaking the *Value Difference Metric* used in the literature is based on a pseudometric. Hence, for clarity we use the name *Value Difference pseudoMetric*. Analogously we use the name *Simplified Value Difference pseudoMetric* instead of the original name *Simplified Value Difference Metric*.

among the chosen values of the attribute BG. For a few cases we present exact computations:

$$\begin{aligned}
&\varrho_{BG}(A,A) = 0,\\
&\varrho_{BG}(A,AB) = |P(D=H|A) - P(D=H|AB)| + |P(D=S|A) - P(D=S|AB)|\\
&\qquad = \left|0 - \frac{2}{3}\right| + \left|1 - \frac{1}{3}\right| = \frac{4}{3},\\
&\varrho_{BG}(A,B) = |P(D=H \mid A) - P(D=H \mid B)| + |P(D=S \mid A) - P(D=S \mid B)|\\
&\qquad = \left|0 - \frac{1}{2}\right| + \left|1 - \frac{1}{2}\right| = 1,\\
&\varrho_{BG}(A,0) = |P(D=H \mid A) - P(D=H \mid 0)| + |P(D=S \mid A) - P(D=S \mid 0)|\\
&\qquad = |0-1| + |1-0| = 2,
\end{aligned}$$

where D, H, S denotes (in this particular example) *Diagnosis*, *Healthy*, *Sick*, respectively; and the prefix '$BG =$' is omitted in the conditions for brevity. For the remaining cases, we present only the final results:

$$\begin{aligned}
&\varrho_{BG}(B,B) = 0, \quad \varrho_{BG}(B,A) = 1, \quad \varrho_{BG}(B,AB) = \frac{2}{3}, \quad \varrho_{BG}(B,0) = 1,\\
&\varrho_{BG}(AB,AB) = 0, \quad \varrho_{BG}(AB,A) = \frac{4}{3}, \quad \varrho_{BG}(AB,B) = \frac{2}{3}, \quad \varrho_{BG}(AB,0) = \frac{2}{3},\\
&\varrho_{BG}(0,0) = 0, \quad \varrho_{BG}(0,A) = 2, \quad \varrho_{BG}(0,B) = 1, \quad \varrho_{BG}(0,AB) = \frac{2}{3}.
\end{aligned}$$

Figure 2.1 graphically presents all the values of the considered attribute and distances between them. For example, it can be seen that values A and *0* are the most distant. It relates to the fact that the corresponding distributions are entirely different (in fact for value A decision is always *Sick* and for *0* always *Healthy*).

Different variants of this pseudometric have been successfully used previously (see e.g. [24, 44, 187]).

It is possible to choose other VDM-based distance functions (see e.g. [213, 219]). In the thesis, experiments carried out for the RIONIDA algorithm with some other variants of pseudometrics are reported too (see Subsection 5.5.5).

2.2.3 Pseudometrics use in the thesis

In the thesis, pseudometrics are used for two reasons. The first one is related to the grouping of values of attributes (see Section 3.2). For each attribute, a relevant group of values for the attribute is expressed as a ball using the respective pseudometric.

The second one concerns searching for nearest neighbours of objects (see Subsection 2.3.3, Section 3.3). Thus, a relevant pseudometric for object space should be provided. First, pseudometrics for single attributes are introduced. Next, is provided an aggregation function which defines pseudometric on the space of objects from given pseudometrics for all attributes.

For these two reasons, we use the following definition.

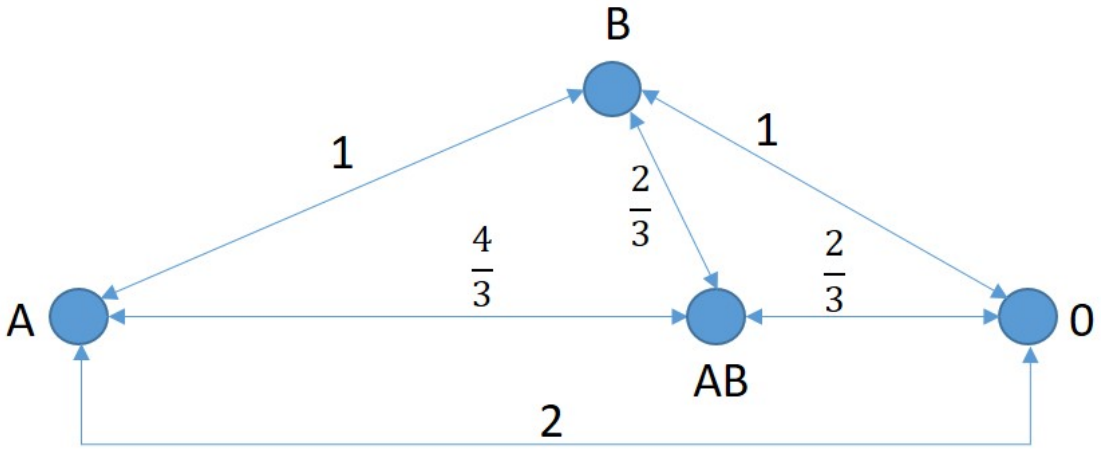


Figure 2.1: Graph representing distances between values of attribute *BloodGroup* induced from training set given in Table 2.1. Vertices correspond to values of the considered attribute, i.e. A, B, AB and 0. Arrows correspond to distances between pairs of values. For clarity, arrows representing distances of value to itself (equal to zero) are omitted.

Definition 2.3. *Let* $(\mathbf{X}, A, d)$ *be a decision system (see Definition 2.1) and for any attribute* $a \in A$, ϱ_a *be a pseudometric on the respective value set* V_a, *i.e. for any* $a \in A$, (V_a, ϱ_a) *is a pseudometric space. We call such an enriched decision system the* pseudometric decision system *and denote it by* $(\mathbf{X}, A, d, \{\varrho_a\}_{a \in A})$.

Additionally, we usually also assume that for a given pseudometric decision system, there is an *aggregated pseudometric* on the object space defined from the individual pseudometrics for attributes and we denote it as $Agr(\{\varrho_a\}_{a \in A})$, i.e. for $\varrho = Agr(\{\varrho_a\}_{a \in A})$, $(\mathbf{X}, \varrho)$ is a pseudometric space.

It should be noted that for numerical attributes, the natural Euclidean metric and additionally the natural ordering of real numbers are used for grouping numerical values into intervals (see Subsection 3.2.1). Thus, for the task of grouping attribute values, pseudometrics for symbolic attributes should be only provided (see, e.g. Definition 3.2 in Subsection 3.2.2).

By default, we assume that the aggregated pseudometric is simply defined as the sum of individual pseudometrics (see Equation 2.1). Rarely, we also use in the thesis its weighted version (see Equation 3.3). The default pseudometric for symbolic attributes is SVDM. Taking this into account, a default resulting pseudometric for measuring the distance between objects combines (according to Equation 2.1) Euclidean metrics on $\mathbb{R}$ for numerical attributes (see Equation 2.2) and SVDM pseudometrics for symbolic attributes (see Equation 2.4). One may say that it combines the normalised city-block metric for the group of numerical attributes and the SVDM pseudometrics for symbolic attributes. We call it *City And Simplified Value Difference pseudoMetric* (CSVDM). Such an aggregation of pseudometrics was used previously in the literature (see e.g. [53]). Rarely, we also use in the thesis discrete metric (see Equation 2.3) for symbolic attributes. In this case, one may say that the resulting metric for measuring the distance between objects combines the normalised city-block metric for the group of numerical attributes and Hamming

metric (see Subsection 2.2.2) for the group of symbolic attributes. We call it *City And Hamming Metric* (CHM).

2.3 Selected methods in machine learning

In this section, we discuss machine learning methods directly related to the thesis, i.e. rule-based, instance-based, and lazy rule learning. In particular, we formalise them for specific approaches used and further developed in the thesis (see Chapters 3, 4). Moreover, we present a crucial formal result connecting these approaches. This result is also further developed in the thesis (see Subsection 3.2.2).

2.3.1 Rule-based methods

One of the critical ML techniques is the induction of rule sets (see e.g. [26, 72, 87, 146, 164, 184, 189]). Its importance follows from the fact that knowledge representation in the form of rules is well understandable by a human. We are interested in decision rules which indicate what decision should be taken in a perceived situation. In the most general form, *decision rules* are of the form

$$if\ \varphi\ then\ \psi\ ,$$

where φ is called the premise of the rule and ψ its consequence. ψ is a formula determined by a decision attribute d.

Rule induction algorithms induce decision rules from a training set. We define what kind of rules we admit and in consequence what kind of rules we search for. We consider decision rules with premises consisting of a conjunction of *elementary conditions* and their consequences indicating the specific decision. Each elementary condition describes a set of values of the attribute. Informally, it is of the form $a \in V$, where $V \subseteq V_a$. First, we define how such sets V of values can be expressed over a formal language together with semantics (meaning) of expressions from this language in the power set of attribute codomain.

Definition 2.4. *Let* $\mathbb{D} = (\mathbf{X}, A, d)$ *be a decision system (or* $\mathbb{D} = (\mathbf{X}, A, d, \{\varrho_a\}_{a\in A})$ *be a pseudometric decision system). The* description of any elementary set *for symbolic attributes* $a \in A_{sym}$ *is one of the following forms:*

$$\emptyset \tag{2.5}$$

$$\{v\},\ where\ v \in V_a, \tag{2.6}$$

$$V_a, \tag{2.7}$$

$$B(c, r),\ when\ \mathbb{D}\ is\ pseudometric\ decision\ system\ and\ where\ c \in V_a,\ r \in \mathbb{R}, r \geq 0. \tag{2.8}$$

The description of elementary set *for decision attribute* d *is of the form 2.6, where attribute* a *is substituted by* d.

The description of elementary set *for numerical attributes $a \in A_{num}$ is of the following form:*

$$\emptyset \tag{2.9}$$

$$[b, e], (b, e], [b, e), (b, e), \text{ where } b, e \in \mathbb{R} \text{ are such that the corresponding interval between points } b \text{ and } e \text{ is included in } V_a. \tag{2.10}$$

The semantics of any description of the elementary set for attribute $a \in A \cup \{d\}$ is defined as a subset of V_a as follows[6]*:*

$$\begin{aligned}
||\emptyset||_{\mathbb{D}} &= \emptyset, \\
||\{v\}||_{\mathbb{D}} &= \{v\} \text{ (it is called the singleton set)}, \\
||V_a||_{\mathbb{D}} &= V_a \text{ (it is called the value set of } a), \\
||[b, e]||_{\mathbb{D}} &= [b, e], \\
||(b, e]||_{\mathbb{D}} &= (b, e], \\
||[b, e)||_{\mathbb{D}} &= [b, e), \\
||(b, e)||_{\mathbb{D}} &= (b, e), \\
||B(c, r)||_{\mathbb{D}} &= \{w \in V_a : w \in B(c, r)\} = \{w \in V_a : \varrho_a(c, w) \leq r\} \text{ (it is called the ball set)}.
\end{aligned}$$

Now, we define the representation of the elementary conditions (in a language) and their semantics.

Definition 2.5. *Let $\mathbb{D} = (\mathbf{X}, A, d)$ be a decision system (or $\mathbb{D} = (\mathbf{X}, A, d, \{\varrho_a\}_{a \in A})$ be a pseudometric decision system).*

The elementary condition *for attribute $a \in A \cup \{d\}$ has the form:*

$$a \in V,$$

where $a \in A$ and V is a description of elementary set for attribute a. Its semantics is defined as follows:

$$[[a \in V]]_{\mathbb{D}} = \{x \in \mathbf{X} : a(x) \in ||V||_{\mathbb{D}}\}$$

The semantics of $a \in V$ may be restricted to subsets of $\mathbf{X}$, e.g. to the training set, trnSet, i.e. $[[a \in V]]_{\mathbb{D}} \cap trnSet$ denoted as $[[a \in V]]_{trnSet}$ The elementary condition t is satisfied by an example x (or, in short, $t(x)$ is satisfied) if $x \in [[t]]_{\mathbb{D}}$.

Please note that $||V||_{\mathbb{D}}$ denotes a subset of the attribute value set of a given attribute, while $[[a \in V]]_{\mathbb{D}}$ denotes a subset of $\mathbf{X}$. Each elementary condition is of the form: $a \in V$, where $||V||_{\mathbb{D}} \subseteq V_a$ and $||V||_{\mathbb{D}}$ is

- a singleton set for decision attribute (see set description 2.6 and its semantics),

[6] For simplicity we do not distinguish between symbols denoting values and values themselves.

- a proper interval for the numerical attribute (see set description 2.10 and its semantics), and
- a singleton set, value set of an attribute or a ball set for the symbolic attribute (see set descriptions 2.6, 2.7, 2.8, respectively and their semantics).

The elementary condition is satisfied for a given object if the value of the concerned attribute on this object belongs to the set given by its description. Conditions of the form $a \in \{v\}$, where $v \in V_a$ are also written as $a = v$. Conditions of the form $a \in V_a$ which are always true (i.e. the set of objects satisfying the condition is equal to the set of all objects in the considered universe), also written as $a = *$, are called *trivial*.

Finally, we define the semantics and the syntax for expressing premise and consequence of the decision rules.

Definition 2.6. *Let $\mathbb{D} = (\mathbf{X}, A, d)$ be a decision system (or $\mathbb{D} = (\mathbf{X}, A, d, \{\varrho_a\}_{a \in A})$ be a pseudometric decision system). A* decision rule *is an expression of the form*

$$if\ t_1 \wedge t_2 \wedge \ldots \wedge t_m\ then\ d = v,$$

where m is the number of attributes, t_i is an elementary condition for an attribute a_i for $i = 1, 2..., m$, $v \in V_d$.

The semantics of the premise of the rule r, φ is defined as follows:

$$[[\varphi]]_{\mathbb{D}} = [[t_1 \wedge t_2 \wedge \ldots \wedge t_m]]_{\mathbb{D}} = [[t_1]]_{\mathbb{D}} \cap [[t_2]]_{\mathbb{D}} \ldots [[t_m]]_{\mathbb{D}}$$

The premise of the rule r, φ, is satisfied by example x (or x satisfies φ) if $x \in [[\varphi]]_{\mathbb{D}}$. In this case, example x is said to match *the rule r, and r is said to* cover *x.*

The single rule is a classifier which classifies examples covered by that rule to the decision class indicated by the rule's consequence. Ideally, we could search for rules $if\ \varphi\ then\ d = v$ such that $[[\varphi]]_{\mathbb{D}} \subseteq [[d = v]]_{\mathbb{D}}$. However, the semantics of $[[d = v]]_{trnSet}$ is available only. Hence, we induce rules for $trnSet$ and assume that the inclusion extends on $\mathbf{X}$. Moreover, we search for rules covering as many as possible examples.

Usually, while presenting the decision rule, trivial conditions are omitted[7]. The commonly used conditions for symbolic attributes are equations $a = v$, while for numerical attributes conditions are specified by interval inclusions, e.g.:

$$if\ a_1 = 2 \wedge a_3 \in [3, 7] \wedge a_6 = 5\ then\ d = 1.$$

However, for symbolic attributes we use a more general condition such as $a \in V$ (see Definition 2.5), which is introduced to extend the notion of the singleton sets to the ball sets specified by form 2.8 in Definition 2.4 and its semantics. If the data set of the considered problem contains some numerical attributes, then the relevant intervals can be obtained by applying discretisation. Discretisation transforms decision system into a new one in such a way that numerical values are grouped into relevant intervals covering the whole attribute domain. Consecutive intervals induced from the original

[7] In fact, in the description of rules only non-trivial conditions are used. We use trivial conditions only to make the notation simpler.

table are mapped into successive numbers representing values of the discretised attribute in a new decision system (see e.g. [161]).

For any decision rule r, we denote by $t_i(r)$ the i-th condition t_i from Definition 2.6 for rule r; we denote by $t_a(r)$ for $a \in A$ the condition t_i from Definition 2.6 for rule r corresponding to attribute a.

In the thesis, we consider three kinds of decision rules relative to the admissible elementary conditions used in Definition 2.6. Below we specify three possibilities of admissible elementary conditions used in Definition 2.6. They specify three kinds of decision rules, and in consequence, three sets of decision rules.

Definition 2.7. *Let* $(\mathbf{X}, A, d)$ *be a decision system.*

For the data sets with only symbolic attributes the set of simple rules *denoted as SimRules is the set of all rules from Definition 2.6 in which the only admissible elementary conditions in the premise of the rule are from set descriptions 2.6, 2.7, i.e. elementary conditions are* $a = v$ *for* $v \in V_a$*; and* $a = *$.

The set of combined rules *denoted as CombRules is the set of all rules from Definition 2.6 for which the only admissible conditions in the premise of the rule are from set descriptions 2.6, 2.7, 2.10, i.e. elementary condition for symbolic attributes are as in the definition of SimRules and for numerical attributes are of the form* $a \in I$*, where* I *is a proper interval description.*

Definition 2.8. *Let* $(\mathbf{X}, A, d, \{\varrho_a\}_{a\in A})$ *be a pseudometric decision system.*

Suppose that for all symbolic attributes $a \in A_{sym}$ *there is given a specific value* $c_a \in V_a$*. The set of* general rules *denoted as GenRules* $\left(\{(\varrho_a, c_a)\}_{a\in A_{sym}}\right)$ *or simply GenRules (whenever pairs* (ϱ_a, c_a) *are clear from the context or irrelevant due to generality) is the set of all rules from Definition 2.6 for which the only admissible elementary conditions in the premise of the rule contain set descriptions (i) as in the definition of CombRules for numerical attributes and (ii) specific form of 2.8, i.e.* $B(c, r)$*, where* $c = c_a$*,* $r = \varrho_a(c_a, v)$*,* $v \in V_a$ *for symbolic attributes* $a \in A_{sym}$*.*

The definition of *general rules* will become more clear after reading Section 3.2, where it is used.

Definition 2.9. *A rule* r *with the consequent* $(d = v)$ *is* consistent *with a set of objects* $X \subseteq \mathbf{X}$ *(sometimes we write simply* consistent *whenever set* X *is clear from the context) if for each object* $x \in X$ *whenever* x *matches the rule* r *the decision of the rule is identical with the decision of the object, i.e.* $d(x) = v$*.*

A rule r *is* inconsistent *if it is not consistent.*

Usually, in the above definition we use as a set X, the set of training objects. A rule consistent with the training set classifies correctly all the training examples covered by that rule.

Now we define the notion of maximality of rule.

Definition 2.10. *Let* r_1 *and* r_2 *be rules. We say that a condition* $t_i(r_2)$ *is* more general than *(or is* implied by*) a condition* $t_i(r_1)$*, in symbols* $t_i(r_1) \Rightarrow t_i(r_2)$*, if* $||V_1||_{\mathbb{D}} \subseteq ||V_2||_{\mathbb{D}}$ *holds, where* $t_i(r_1)$ *is of the form* $a_i \in V_1$ *and* $t_i(r_2)$ *is of the form* $a_i \in V_2$*.*

We say that a rule r_2 is more general than *(or is* implied by*) a rule r_1, and denote it by $r_1 \Rightarrow r_2$ if it has identical consequents, i.e. $d(r_1) = d(r_2)$ and each condition $t_i(r_2)$ is more general than condition $t_i(r_1)$ (for $i = 1, 2, \ldots, m$).*

A consistent rule r with a training set $trnSet$ is maximally general *(relative to this training set and a given set of rules) if there is no rule in this set of rules more general than r which is different from r and consistent with $trnSet$.*

Definition 2.11. *For a given set of admissible rules $Rules$, a training set $trnSet$ we define the* set of maximally general rules $MaxRules(Rules, trnSet)$ *to be equal to the set of all rules $r \in Rules$ consistent with $trnSet$ and maximally general.*

In the thesis, we consider only sets of rules $Rules$ in Definition 2.11 equal to one of three sets: $SimRules$, $CombRules$, $GenRules$ from Definitions 2.7 and 2.8. If the sets $Rules$ and $trnSet$ are obvious from the context, we write $MaxRules$ instead of $MaxRules(Rules, trnSet)$. Also, we write $MaxRules$ in general case, i.e. $MaxRules$ denotes any one of the three mentioned cases: $MaxRules(SimRules, trnSet)$, $MaxRules(CombRules, trnSet)$ or $MaxRules(GenRules, trnSet)$.

From the knowledge discovery perspective, the important problem is to compute all rules (matched at least by one training example) that are maximally general and consistent with a training set.

Let us first consider $MaxRules(SimRules, trnSet)$. In this case, a consistent rule is maximally general in a training set $trnSet$ if for each non-trivial condition replacement of that condition with trivial condition makes the rule inconsistent with the training set $trnSet$. Hence, maximally general rules are those which have minimal lengths, where the length of the rule is the number of non-trivial conditions in it. Thus the problem here is to find the complete set of consistent and minimal decision rules (see e.g. [184]). Among different aspects, such a set of rules is also essential because it relates to the minimal description length principle (see e.g. [182]). Algorithms for computing all minimal rules are very time consuming, especially when the number of training objects or attributes is significantly large. This is because the size of the $MaxRules$ set can be exponential concerning the size of the training set (see e.g. [158]). In practice, approximation algorithms are often applied to obtain the rule set that is not necessarily complete (see e.g. [16]). There are also other approaches to induce a set of rules, which cover the input examples using, e.g. the smallest number of rules (see e.g. [86]). However, in the thesis, we focus on the complete $MaxRules$ set.

Now, let us consider $MaxRules(CombRules, trnSet)$. In this case, additionally we have numerical attributes for which maximally general intervals are searched. Searching for maximally general rules for numerical attributes relates to the problem of discretisation. A partition of discretisation is consistent if each interval covers only objects with the same decision. For more details on discretisation, the readers are referred to [159, 161].

It should be noted that the problem of searching for a consistent partition with the minimal number of cuts is NP-hard (see e.g. [161]). It shows that the problem of discretisation from the global point of view is a complex task. We will show in Subsection 3.2 that it is in a sense possible to overcome this problem if one focuses on a local fragment of the universe instead of the whole universe. It occurs in case of the presented lazy rule induction algorithm (see Algorithm 2 or Algorithm 4).

Now, let us consider $MaxRules(GenRules, trnSet)$. In this case, we additionally search for relevant grouping of values for symbolic attributes. It relates to the problem of partition of symbolic attributes. Formally the partition over an attribute a is any function $P_a : V_a \to \{1, \ldots, m_a\}$. The problem of searching for a consistent family of partitions with the minimal $\sum_{a \in A} |P_a(V_a)|$ is NP-hard (see e.g. [160]). We overcome this because of two reasons. First, we limit the number of possible groupings of values of any attribute (from 2^n to n^2, where n is the number of values for an attribute). Second, we use lazy rule induction (see Section 3.2).

Rules induced from training examples are then used to classify objects. For a given test object, the subset of rules matched by the object is selected. If the object matches only rules with the same decision, then the decision predicted by those rules is assigned to the example. If the test object matches the rules corresponding to different decisions, the conflict has to be resolved (see e.g. [147]). A common approach is to use a measure for conflict resolution, and decision with the highest value of the measure is selected. In the thesis, we focus on the commonly used measure that is presented below.

Definition 2.12. *Suppose training set trnSet, test example tst and MaxRules are given. Then we define*

$$Strength(tst, v) = \left| \bigcup_{r \in MatchRules(tst,v)} supportSet(r) \right|, \tag{2.11}$$

where v denotes the v-th decision ($v = 1, \ldots, n_d$), $supportSet(r)$ is a set of training examples matching the rule r, $MatchRules(tst, v)$ is a subset of $MaxRules$, whose premise is satisfied by tst and the consequent is a decision v.

The measure *Strength* counts the number of training examples covered by the maximally general rules with the decision v and covering a test example *tst*.

The classifier based on maximally general rules with the measure *Strength* as a strategy for conflict resolution predicts the decision that is the most frequent in the set of training examples covered by rules matched by a test example, i.e.:

$$decision_{MaxRules}(tst) = \arg\max_{v \in V_d} Strength(tst, v). \tag{2.12}$$

As mentioned previously, the limitation of this approach lies in the fact that computing $MaxRules$ is very time-consuming.

2.3.2 Lazy rule learning for symbolic attributes

Another approach can be based on a construction of algorithms that do not require calculating the set of decision rules before classifying new objects. These are *lazy learning* (or *memory based learning*) algorithms. An example of such an algorithm for the case of *SimRules* is presented in [15]. It generates only decision rules relevant for a new test object and then classifies it like algorithms generating rules in advance. It uses a technique that computes the measures from Equation 2.11 for every test object without computing all maximally general rules ($MaxRules$).

First, we define *simple local decision rule*, denoted by $s\text{-}rule(tst, trn)$, where tst, trn are the distinguished objects. This name corresponds to the set of simple rules, denoted by $SimRules$ (in the following proposition we show their actual relationship).

Definition 2.13. *For any test object tst and any training object trn, we define a* simple local decision rule *(for short* s-rule*), denoted by $s\text{-}rule(tst, trn)$, the decision rule with the decision $d(trn)$ and the following conditions t_a for each symbolic attribute a:*

$$t_a = \begin{cases} a = a(trn) & \text{if } a(tst) = a(trn) \\ a = * & \text{if } a(tst) \neq a(trn), \end{cases}$$

Let us recall that $a = *$ denotes the trivial condition $a \in V_a$. A local decision rule is defined to ensure that both trn and tst objects satisfy the rule and it is maximally specific (the number of trivial conditions is minimal; or inversely, the number of non-trivial conditions is maximal). We have the following crucial relation between s-rule and maximally general consistent rules from $SimRules$:

Theorem 2.2. *[15][8] The rule $s\text{-}rule(tst, trn)$ for a test object tst and a training object trn is consistent with the training set $trnSet$ if and only if there exists a rule in the set $MaxRules(SimRules, trnSet)$ covering objects tst and trn.*

It means that for $MaxRules(SimRules, trnSet)$ for any test example tst, any decision $v \in V_d$ computing the value of measure $Strength(tst, v)$ from Equation 2.11 is equivalent to computing the number of training examples $trn \in trnSet$ such that $d(trn) = v$ and rule $s\text{-}rule(tst, trn)$ is consistent with $trnSet$. This is realised by the *simple lazy rule induction algorithm for symbolic attributes* (LAZY) presented below (see Algorithm 2).

Algorithm 1: isConsistent(r, $verifySet$)

Input: a rule r : if α then $d = v$, set of examples $verifySet$
Output: true if rule r is consistent with $verifySet$, false otherwise

```
1 begin
2     foreach trn ∈ verifySet do
3         if d(trn) ≠ v and trn satisfies α then
4             return false
5         end
6     end
7     return true
8 end
```

The function $isConsistent(r, verifySet)$ checks if a decision rule r is consistent with a $verifySet$. For every training object trn, Algorithm 2 constructs the rule $s\text{-}rule(tst, trn)$ based on the examples tst and trn. Then it checks whether the

[8] The original formulation of this proposition was different. However, this formulation in the considered case of $SimRules$ is equivalent to the original proposition. Such formulation allows us to show a more direct relationship between local rules and $MaxRules$ and algorithms based on these two types of rules.

Algorithm 2: LAZY(tst, $trnSet$)

Input: test example tst, training set $trnSet$
Output: predicted decision for tst

```
1  begin
2      foreach decision v ∈ V_d do
3          supportSet(v) = ∅
4      end
5      foreach trn ∈ trnSet do
6          v = d(trn)
7          if isConsistent(s-rule(tst, trn), trnSet) then
8              supportSet(v) = supportSet(v) ∪ {trn}
9          end
10     end
11     return arg max_{v∈V_d} |supportSet(v)|
12 end
```

rule $s\text{-}rule(tst, trn)$ is consistent with the remaining training examples, i.e. if all the training examples satisfying the left-hand side of $s\text{-}rule(tst, trn)$ are labelled by the same decision as the considered training example trn. If the rule $s\text{-}rule(tst, trn)$ is consistent, then the training example trn is added to the support set of the relevant decision. Finally, the algorithm selects the decision with the support set of the highest cardinality. As it was mentioned above from Theorem 2.2 we have:

Corollary 2.3. *Let $trnSet$ be a training set. For any test object tst, and the classifier from Equation 2.12 with $MaxRules = MaxRules(SimRules, trnSet)$, we have*
$LAZY(tst, trnSet) = decision_{MaxRules}(tst)$.

Comparison of the LAZY algorithm to the algorithm based on maximally general rules allows us to conclude that the LAZY algorithm considers only the decision rules that can be involved in the classification of a given test object.

The time complexity of the procedure *isConsistent* checking whether a decision rule r is consistent with $trnSet$ is $O(mn)$, where $n = |trnSet|$, $m = |A|$ (i.e. n is the number of training examples and m is the number of conditional attributes). In consequence, time complexity of the LAZY algorithm (see Algorithm 2) checking the consistency of the local simple decision rule based on all possible training examples is $O(mn^2)$. In the case of classifying many test objects, this expression should be multiplied by the number of test cases. This is far more efficient than generating the set $MaxRules(SimRules, trnSet)$ in advance, which can be exponentially large relative to n.

The limitation of this approach is that it works only with symbolic attributes and does not allow to group symbolic attributes (i.e. for symbolic attributes it allows only the use of equality descriptor). In Section 3.2, we present the RIA algorithm, a version of the LAZY algorithm extended to the case of numerical attributes and generalised for symbolic attributes.

Another limitation of this approach is that for larger data sets this time complexity is still too high to be used in practice. In Section 3.3, we present a modification of

the RIA algorithm, which also applies to the LAZY algorithm as a subcase of RIA. This modification gives a further reduction of time complexity without decreasing the classification quality in practical applications.

2.3.3 Instance-based learning

In the previous subsection, an example of lazy learning algorithm is presented. In general *lazy learning* (for supervised learning) refers to a class of procedures which simply store learning examples (thus called also *memory based learning*), and when classification is requested, it induces the decision directly from the stored data (see e.g. [6, 149]). It is in contrast to *eager learning*, mostly occurring in ML, which refers to a family of algorithms generalising relevant patterns from specific instances e.g. rule-based methods induce rules to be later used for classification.

A typical example of lazy learning is *instance-based learning*. It can be described by a simple principle stating that *similar instances have similar class labels* (see e.g. [5, 7])[9]. Most instance-based learning classifiers can be characterised by four components: 1. similarity (or distance) measure, 2. number of neighbours to consider (from one to all neighbours), 3. function of weight for neighbours (e.g. equal weights or weights depending on the distance from the test object), 4. conflict resolution (e.g. majority vote of the k nearest neighbours) (see e.g. [182]).

The commonly used instance-based algorithm is the k nearest neighbours classification algorithm (kNN). It is based on the assumption that for classification of a given test example it is enough to use (training) examples that are sufficiently close to this example. Hence, test examples are classified using, e.g. the decision most common in the set of k nearest neighbours from the training set.

For $k = 1$, it returns the decision of the training example most similar (assuming that exactly one such example exists) to the test case (according to the given pseudometric ϱ). That is:

$$decision_{1NN}(tst) = d(trn^{\star}), \text{ where } trn^{\star} = \underset{trn \in trnSet}{\arg\min}\, \varrho(tst, trn).$$

For general case, the kNN method works as follows. It selects k nearest neighbours to the example tst according to the given pseudometric ϱ[10].

Definition 2.14. *For training set $trnSet$ and test example tst we define $N(tst, trnSet, k, \varrho)$ as the set of k training examples that are most similar to tst according to distance function ϱ. In the case when more than one example has the same distance from the object tst to the k-th nearest example, all of them are added to $N(tst, trnSet, k, \varrho)$ (then the set $N(tst, trnSet, k, \varrho)$ contains more than k examples)*[11].

[9] More generally, instance-based learning can refer to Case Based Reasoning (see e.g. [175]), i.e. a family of techniques which solve unseen problems based on the solutions of similar problems perceived in the past.

[10] To be consistent with other places in the thesis we focus on pseudometrics also here. See also beginning of Section 2.2 for other possibilities.

[11] Such solution is used in the proposed algorithms in the thesis. Thus we also use it for the kNN algorithm.

For short, we also write $N(tst, k)$ whenever parameters $trnSet$ and ϱ are clear from the context (or irrelevant due to generality). We also write N whenever all parameters are clear from the context (or irrelevant due to generality).

The decision with the majority of examples in this neighbour set is selected as the final decision.

$$decision_{kNN}(tst) = \arg\max_{v \in V_d} |\{trn \in N(tst, trnSet, k, \varrho) : d(trn) = v\}| \tag{2.13}$$

We can represent Equation 2.13 in the form presented in Algorithm 3.

Algorithm 3: kNN(tst, $trnSet$, k, ϱ)

Input: a test example tst, training set $trnSet$, positive integer k, pseudometric ϱ
Output: predicted decision for tst

```
1  begin
2      neighbourSet = N(tst, trnSet, k, ϱ)
3      foreach decision v ∈ V_d do
4          supportSet(v) = ∅
5      end
6      foreach trn ∈ neighbourSet do
7          v = d(trn)
8          supportSet(v) = supportSet(v) ∪ {trn}
9      end
10     return arg max_{v∈V_d} |supportSet(v)|
11 end
```

For determinism, there should also be specified a tie-breaking procedure.

2.4 Imbalanced data

Two important questions should be considered here: *What does it mean that data are imbalanced?; What are the factors of the difficulty of imbalanced data?* These topics refer to foundations of imbalanced learning. This topic has attracted extensive research. The overview studies which merit our special attention include: [152, Chapters 2-3], [102, Chapter 2], [139], [209].

These two questions, which are strongly related, are discussed below.

2.4.1 Basic definition of imbalanced data and its drawbacks

Technically speaking, any data set considered as an input for the classification problem in which examples of one class significantly outnumber the examples of the other one can be considered imbalanced (see e.g. [101, 139]). The ratio of the number of examples from the majority class and the minority class is called *imbalance*

ratio. The imbalance ratio can range from 2:1 (moderate level of class imbalance) to *imbalance ratios* above 1000:1 (extreme level of class imbalance; one thousand more objects from the majority class than from the minority class). For real-life examples with different *imbalance ratios* see e.g. [118, 125, 224].

Experiments show that in some cases the growth of imbalance ratio leads to declining of the classification quality (see e.g. [102, Chapter 2], [211]). Thus sometimes this factor can represent to a degree the difficulty of classification.

However, the difficulty of learning from imbalanced data is concerned not only of the imbalance ratio between classes but mainly of the *data complexity* (see e.g. [110, 139, 152]). We discuss this topic in the next subsection. Let us consider a simple illustration that data complexity and the imbalance ratio factor do not coincide. For instance, for very complex data set, one can multiply one chosen example from the minority class to obtain the data set with equal representation of both decision classes[12]. In this case, we obtain the data set with the imbalance ratio equal to 1:1, but the difficulty of the data set is similar to the original data set. If we take into account this fact together with topics discussed in the next subsection one can see that difficulty of data set cannot be measured only by imbalance ratio and doing this sometimes can be misleading (see e.g. [102]).

Let us also note that the imbalance ratio is defined on the data we possess. If the data are representative for the underlying distribution, what is usually assumed, we have a good estimation of this parameter. Otherwise, the estimated imbalance ratio can be misleading. In consequence, even balanced data sets could be recognised as imbalanced and vice versa. (see e.g. [102]).

2.4.2 Different factors of the difficulty of imbalanced data

As it was mentioned above, the imbalance ratio taken separately is insufficient to measure the difficulty of learning from imbalanced data. Generally speaking, the difficulty of imbalanced data is embodied in the complex structure of the minority class concept.

The literature distinguishes several factors which make the learning from imbalanced data a challenging task (see e.g. [11, 110, 139, 210, 211]). Among them are:

- selection of relevant performance measure,
- relevant representation,
- data decomposition leading to *small disjuncts*,
- overlapping between the classes,
- presence of outliers or noisy examples,
- imbalance ratio,

[12] If we proportionally multiply all examples from the minority class we, in fact, have simple sampling method which still does not solve the problem of data complexity (see Subsection 2.5.1). It should be noted that our intention was to make the example as simple as possible.

- the absolute number of examples.

As it can be noticed, the factor of the imbalance ratio is usually combined with other factors. All these factors are briefly discussed below. Let us notice that the first two factors are related to the primary components in any data-mining algorithm (see e.g. [59]).

Selection of relevant performance measure

The main task in solving the classification problem is to understand the given data mining problem. This includes understanding what does it mean that a classifier performs well on a given domain. The *performance measure* (see also Subsection 2.6.1) is used to express this precisely. It is important is to find a performance measure relevant to the considered problem (see e.g. [59]). Although this issue is important for any classification task, it is especially important for imbalanced learning problem. If such performance measure is available, it should be embedded in the classifier induction algorithm (see e.g. [102]). However, for many imbalanced learning problems, such performance measure is unknown, and an approximation of the ideal performance measure is used. Examples of general performance measures are presented in Subsection 2.6.1. However, we want to stress here that if one is inducing a classifier by using optimisation relative to an ad hoc selected performance measure, then with a high chance the resulting classifier may differ from the one, the user is interested in.

Specification of the exact requirement for the classifier is one of the important tasks to accurately solve the imbalanced learning problem. This is a much harder task than in case of balanced data where the widely used accuracy measure can be usually selected as the relevant performance measure to start with.

Relevant representation

Relevant representation (relevant space of objects) can influence imbalanced learning problem. Selecting relevant attributes is generally important and hard problem in data mining (see e.g. [27, 93, 134]). This is especially important for imbalanced data (see e.g. [197]). It was shown for the real-life problems that feature extraction can be of much higher importance than selecting proper learning method (see e.g. [201]). This is particularly important for high-dimensional and imbalanced data sets (see e.g. [150]). Searching for relevant features should not be considered separately, but in correlation with other features (see e.g. [92]).

Furthermore, using the relevant similarity measure between two entities can be crucial for some ML methods (e.g. for kNN methods). Finding the relevant similarity measure is an important and hard problem in data mining (see e.g. [12, 28, 99, 107]). By using the relevant similarity measure, one can appropriately group entities labelled by the same decision. This can change the difficulty of the classification task for chosen methods.

Thus, selecting the relevant attributes and/or the relevant similarity measure for grouping objects is crucial for solving the classification task. This can significantly change the nature of the problems listed below.

Data decomposition leading to small disjuncts

Suppose we have fixed a language for expressing concepts (e.g. rules, similarity relations for grouping similar objects in clusters); in that case, one can group examples with the same decision into some regions (clusters). The sum of these regions defines the target concept. All the regions used to describe the target concept can be regarded as subconcepts. For example, if the target concept is related to the presence of cancer (the minority class), then the subconcepts can correspond to different types of cancer. Among those subconcepts in the minority class, there can be subconcepts of different cardinality. The problem is that some or even many of those subconcepts can be supported with relatively few examples only. In the discussed example, there could be a rare subconcept of some specific type of cancer.

This situation is known in the literature as *within-class* imbalance (see e.g. [209]). The imbalance defined in the previous subsection is also called *between-class* imbalance.

The name *small disjuncts* derives from the classifiers which represent concepts as a function of conjunctions of conditions for attributes (e.g. rule-based classifiers, tree classifiers). For example, a single rule in rule induction algorithms is represented as a conjunction of conditions for attributes (see Subsection 2.3.1). Final classifier, in the simplest case, can be represented as a disjunction of all conjunctions used in rules describing the positive class. It is known in general that for the systems representing concepts by several disjuncts of conjuncts there occur relatively many conjuncts that have small coverages (see e.g. [103]). Such disjuncts, which cover a few training examples, are called small disjuncts.

The problem with small disjuncts is that they cause much higher error rate than large disjuncts. Moreover, this property is preserved in overall, i.e. finally small disjuncts lead to much higher error rate than the large ones (see e.g. [103]). For balanced data, special *statistical tests* for eliminating small disjuncts can be used (see e.g. [103]). However, in this way for imbalanced data truly important subconcepts could be eliminated. Moreover, when only a small number of examples supporting these subconcepts is available, the statistical tests could be of small significance (see e.g. [103]). Experimental results show that indeed for different classifiers errors are concentrated over smaller disjuncts (see e.g. [210]).

This justifies how important it is to properly find and describe these regions in the target concept. Small disjunct can relate both to the minority class and the majority class, however usually this problem relates to the minority class.

Overlapping between classes

Generally, the imbalanced learning problem is related to separating the minority class from the majority class. If there exist patterns expressed in the selected language which properly discriminate one class from another, the learning task is relatively easy. In such case, usually very advanced method is not needed to solve the classification task, and the value of the imbalance ratio does not influence the final performance of classification. However, if those patterns overlap more and more, the learning task becomes increasingly difficult (see e.g. [76, 168, 195]).

Presence of outliers or noisy examples

Noise affects all data mining tasks. However, its impact can be severe for the imbalanced learning problem. This is due to the fact that noise especially leads to the inappropriate learning of small subconcepts of the minority class, which, in fact, are of special importance [209].

On the other hand, examples which look like noise can be, in fact, examples of outliers, i.e. proper examples which are not similar to the other examples. Deleting such examples, especially examples from the minority class can lead to incorrect classification of rare test examples which are very important to be classified correctly (see e.g. [4, 23]).

Imbalance ratio

To sum up, what was presented in Subsection 2.4.1 and in the above subsections, the imbalance ratio alone cannot reflect the complexity of the data. However, its high value can enhance the difficulty of the target concept determined by the difficulties presented above. In this sense, the imbalance ratio can only measure one of the factors determining the difficulty of data set.

The absolute number of examples

There is also another topic related to the imbalance ratio. This value shows the relative difference between classes. However, in practice, the absolute number of examples can play an important role in the difficulty of data sets. For instance, for imbalance ratio equal to 100:1, the easier task would be if we have 10 000 positive examples and 1 000 000 negative examples than if we have 10 and 1000 examples, respectively. This is due to the fact that some subconcepts of the minority class in the latter case could be represented only by one or even none example. This problem is referred in the literature as the *absolute rarity* whereas the problem of the high imbalance ratio is referred as the *relative rarity* (see e.g. [102]).

2.4.3 Types of examples indicating the complexity of the data sets

How to describe quantitatively the overall complexity of the imbalanced data sets? It was noticed that some examples in the data set are easy to classify, and some are hard. The intuition is that the more number of hard examples are in the considered data set, the more complex is the data set. Some attempts were made to discriminate different *types of examples* within the minority class (see e.g. [124, 152, 154, 155, 191]). Each of these types relates to a different kind of hardness. In [152], the following types of minority class examples have been identified:

- *safe*,
- *borderline*,
- *rare*,

- *outlier*.

Let us remember that we focus on the types of examples for the minority class. The *safe* examples are those which lay in the interior of the homogeneous regions of the minority class. The *borderline* examples are those which are located close to the boundary between two classes. An *outlier* example is surrounded by examples from the majority class. Any *rare* example is nearly like an outlier, with such difference that in relatively close distance from it there is another example from the minority class.

The identification of these types usually is performed using the neighbourhood of the example. If all or nearly all neighbour examples have the same labels as the minority example, then it is identified as the safe. In case the neighbour examples are nearly equally distributed, the example is identified as the borderline. In case all neighbour examples are from the majority class, the example is identified as an outlier. In case not all but nearly all examples are from the majority class, the example is identified as the rare.

Formally in [152], kNN neighbourhood is used with $k = 5$, and the following distribution of nearest neighbours from the minority and majority classes are used to identify the types of examples:

- safe for 5:0 or 4:1,
- borderline for 3:2 or 2:3,
- rare for 1:4,
- outlier for 0:5.

For other possibilities and more detailed discussion see [152]. The number of examples for all the above types of examples for a data set describe to a degree the complexity of the data set. A big number of safe examples indicate that data set is easy for classification. The more borderline, rare and outlier examples are in a data set, the harder the set is for classification. The big number of outliers indicate the extreme hardness of data set for classification.

2.4.4 Drawbacks of imbalanced data analysis by the standard learning algorithms

The difficulty of imbalanced data analysis can be illustrated by difficulties of such data analysis by standard learning algorithms. Why the quality of standard learning algorithms is low when they are applied to imbalanced data? There are at least four reasons for that.

First, the standard learning algorithms are aiming to maximise classification accuracy expressed by a ratio of the number of correct predictions made by classifier over the total number of predictions made. For imbalanced data, this performance measure is unsatisfactory. Let us consider the previously mentioned example of *Mammography* data set. This data set contains 10 923 non-cancerous (the majority class) and 260 cancerous (the minority class) samples. Let us consider the trivial

classifier selecting the majority class for any patient. Thus, in this example, it would always predict a patient to be healthy. Of course, such classifier is completely useless, but at the same time, it achieves accuracy approximately equal to 98%. For many balanced data sets, such accuracy result would seem excellent. Thus, in this example, it becomes clear that the considered accuracy can lead us to the false conclusion about the classifier's quality. The evaluation of classifier by the accuracy measure becomes inadequate for imbalanced data. The appropriate performance measures for imbalanced data are discussed in Section 2.6. Here, it is worthwhile mentioning that if in the learning phase classifiers are trying to maximise standard performance measures such as the accuracy, then, in the case of imbalanced data, this may lead to a classifier of the low classification quality.

The second problem is related to the fact that most of the standard methods assume or expect on input balanced class distribution. Generally, the classifier is required to achieve a balanced rate of predictive accuracy for both the minority and majority classes. However, standard classifiers while achieving high accuracy for the majority class achieve a rather low accuracy for the minority class (see e.g. [102]). What is the reason for that? We do not intend to give the general answer, but rather some intuition only. Let us consider as an example rule-based classifiers. For imbalanced data, the induced rules usually have different coverage: small for the minority class and big for the majority class. Thus in the conflict resolution between rules matched by a new case, the majority class would usually win.

Third, when the standard learning algorithms are identifying noisy examples and then removing them from the training data, at the same time they may disrupt knowledge encoded in imbalanced data. On the one hand, small clusters of the minority class could be regarded as noise. On the other hand, a few real noisy examples from the majority class not identified as noisy can complicate classification for the minority class (see e.g. [139]).

The fourth problem is that the standard learning algorithms assume equal misclassification costs to all classes. In the example of mammography examinations, standard classifiers would usually treat equally both types of patient's health misclassification. However, in medical practice, these two types of misclassification have very different consequences. If a healthy patient is classified as cancerous, this will lead to some thorough medical examinations (maybe costly). But if the cancerous patient is classified as healthy, it can result in irreversible worsening of her/his health state or even the death. Thus it becomes clear that the misclassification cost in the considered domain should be higher for the minority class than for the majority class. It is required that the classifier predicts correctly most of the truly cancerous patients.

2.5 Existing methods for imbalanced data

Numerous algorithms have been proposed, especially in the last decade, for solving the imbalanced learning problem (see e.g. [95]). In this section, we present only a brief overview of the existing methods. The readers are referred to [95], [139], [152, Chapters 2-3], [102], [101], [195] for more details. We only present details of the algorithms used in the experiments reported in the thesis.

Solutions proposed for the imbalanced learning problem are often grouped into data-level and algorithm-level methods (see e.g. [195]). They are characterised in the following subsections.

2.5.1 Data-level approaches

The main idea behind data-level methods is to transform the original data set into a new one in order to be able to use standard ML techniques. These standard algorithms are usually aiming to maximise the accuracy. For balanced data, when the distribution of both classes is even, it is satisfactory. Methods on data-level try to balance the distribution of both classes to make it like in the case of balanced data. The performance of such methods depends on their ability to solve the problem of data complexity (see Subsection 2.4.2).

The big advantage of these methods is that they are independent of the selected accompanying learning algorithm. Thus, many well-known learning algorithms addressed for balanced data can be used together with any data-level method.

Resampling techniques

Very popular data-level strategies for dealing with imbalanced data are resampling techniques (called also *filters*). Resampling techniques can be divided into three groups:

- over-sampling which increase cardinality of the minority class,
- under-sampling which decrease cardinality of the majority class,
- hybrid methods which combine the previous two approaches.

The simplest over-sampling method is based on random duplication of examples from the minority class (see e.g. [108]). The more sophisticated and very well-known method is `SMOTE` [37]. Due to its simplicity and proven success in various applications, it is a standard benchmark for learning from imbalanced data. This method creates new synthetic examples for all minority class examples. Let us describe over-sampling for one fixed example from the minority class. First, for this example it takes a previously specified number of its nearest neighbours from the minority class (by default 5). Second, depending on the value of the desired over-sampling ratio (e.g. 200%), a relevant number of these neighbours are randomly chosen. Third, for all chosen neighbours, new examples are synthesised between the considered example and the chosen neighbours. This is done by taking random feature values between values of the example and one from the neighbourhood. In this way, new synthesised examples extend the border of the minority class (for more information see [37]).

There are also other sophisticated methods of over-sampling based on `SMOTE` (see e.g. [179, 204]). It should be mentioned here the recent interesting modification of the `SMOTE` algorithm for manifold and synthetic over-sampling (see e.g. [20, 21]). Additionally, in [63], one can find a good overview on variations of `SMOTE` developed during the 15 years since the algorithm was invented. It is also worthwhile mentioning

that there exist SMOTE extensions with adaptive k value related to the data complexity (see e.g. [129, 231]).

Under-sampling methods eliminate some examples from the majority class. The simplest method eliminates randomly chosen examples. A very effective modification of this method is presented in [196]. One of the under-sampling methods is *Edited Nearest Neighbour* (ENN, see [214]). This method discards those majority examples which are close to the minority class. First, it takes 3 nearest neighbours of the considered example from the majority class. If at least two of them are examples from the minority class, then it is removed. Still new approaches for under-sampling to handle imbalanced data are being developed (see e.g. [206]).

Hybrid methods are obtained by the combination of the over-sampling and under-sampling method. As an example of hybrid method, one can consider SMOTE+ENN – first, over-sampling SMOTE is used, and then under-sampling ENN is used (see [14]). It was reported in the literature that this sampling method provides very good results in practice in comparison with many other sampling methods, especially for a small number of instances from the minority class (see e.g. [14]).

The random under- and over-sampling have their drawbacks. Under-sampling may delete potentially useful examples. Over-sampling may enhance the effect of overfitting, especially when it is related to noise in the data.

2.5.2 Algorithm-level approaches

An important direction for solving the imbalanced learning problem is to modify existing learning algorithms to improve their performance for imbalanced data. Usually, this is done by changing bias to the minority class.

For example, there were attempts to modify decision trees to improve performance of the standard C4.5 algorithm for imbalanced learning problem (see e.g. [40, 136]).

In recent years an increasing interest can be observed in areas such as in deep-learning and extreme learning machine. There were also attempts to adapt these approaches to imbalanced data (see e.g. [105, 113, 126, 205, 228], and [52, 173, 227] for deep-learning and extreme learning machine, respectively). However, the relationships of these approaches with imbalanced data still need to be studied more, as it was stated, for example, in the survey [113].

There were attempts to adapt the kNN algorithms to imbalanced learning problem. When one class is dominating in the considered data set, it can be expected that in many regions this class dominates in the neighbourhood. Thus in the regions where two classes overlap standard kNN can be biased towards the majority class. This leads to the misclassification of the minority class. In [135], k-nearest neighbours weighting strategy is proposed for imbalanced learning problem. Training examples are provided with class confidence weights (CCW) according to their probability of attribute values for a given decision class. While standard kNN method uses the probabilities of decision classes in the constructed neighbourhood, this method uses conditional probabilities of decision classes. They show two methods for calculating CCW weights: mixture models for numerical attributes and Bayesian networks for symbolic attributes.

There were attempts to change metric adjusted appropriately for the imbalanced

learning problem. In [123], for each class, its empirical cumulative distribution function of nearest neighbour distances is approximated. It is used in the classification process by calculating the probability that the vector consisting of distances of test example to its k nearest neighbours relates to the considered class. The class with the greatest probability is chosen. In [130], the boundary of the minority class is extended. This is done by selecting minority examples and generalising them to Gaussian balls to represent concepts of the minority class.

In [56], local class distribution is taken into account together with wider region. For the local class distribution, relevant weights are computed based on the wider region. Weights are calculated according to the performance of kNN classifier for the neighbour examples.

There exist some algorithms dedicated to imbalanced data based on modifications of standard rule-based algorithms. First, after [152] (which bases in this aspect on [209]) let us list the typical limitations of standard rule-based approaches: top-down induction technique (favouring general rules); improper evaluation measures used to guide the search; greedy, sequential covering technique (examples covered by rules are not used for generation of other rules); biased classification strategies (conflict resolution is biased towards the majority class). The following approaches partially overcome these limitations: approaches applying less greedy search techniques (see e.g. [90, 193]; see also rule-based one-class learning algorithms in Subsection 2.5.4), approaches based on solutions trying to improve the quality of generalisation of rules for the minority class (see e.g. [157]), approaches using strategies for conflict resolution increasing sensitivity to the minority class (see e.g. [89], [25]), approaches using evaluation measures during rule generation more relevant for imbalanced data (see e.g. [9, 103, 114]), approaches refining rules for the *borderline regions* (i.e. regions containing mainly borderline examples) to better detect the minority class (see e.g. [124, 137, 192]), and approaches combining genetic algorithms with rules to better classify imbalanced data (see e.g. [74, 148]). More detailed overview of these techniques can be found in [152]. As claimed in [152], the general drawback of these approaches is that they do not overcome all the above-listed limitations related to imbalanced data.

Now we briefly characterise two rule-based learning algorithms dedicated to imbalanced data used in the thesis experiments. The first one (MODLEM-C) tries to overcome the problem of biased classification strategy. The second one (BRACID) is of special importance for the thesis since it uses an integrated representation of rules and single instances. In other words, it combines instance- and rule-based approaches analogously to the algorithms presented in the thesis. Moreover, this algorithm is claimed to overcome all the main drawbacks of rule-based learning algorithms in case of imbalanced data contrary to the approaches discussed in the previous paragraph. Therefore, it is especially valuable to compare the algorithm for imbalanced data presented in the thesis (RIONIDA) with BRACID.

MODLEM-C is an extension of the MODLEM algorithm (see [188–190]) allowing strengthening sensitivity to the minority class. For all rules describing the minority class, the rule's strength is multiplied by the same real number. This number is given as a parameter, which is called the *strength multiplier*. This is equivalent to adding duplicates of objects from the minority class to the training set. For more

information, see [89, 90] (and citations given for MODLEM above).

Bottom-up induction of Rules And Cases for Imbalanced Data (BRACID) is a modification of RISE algorithm (see [53]). Analogously to RISE it uses an integrated representation of rules and single instances. It uses the strategy of bottom-up induction of rules from single examples with the specific generalisation by searching for nearest examples to the rule. Using F-measure, it evaluates a generated rule relative to this rule's local recognition of decision classes. It distinguishes a few types of examples (as described in Subsection 2.4.3) and treats them differently. The conflict resolution bases on supports of the nearest rules to the test example. For more information, see [152, 153].

2.5.3 Cost-sensitive learning

Cost-sensitive learning assumes that there is given a cost matrix describing the costs of misclassifying one class as another. The goal of learning is to construct a classifier minimising the overall cost on the training data set. Research shows that the methodology of cost-sensitive learning can be naturally applied to the imbalanced learning problem (see e.g. [38, 209]). The main idea behind the cost-sensitive learning for imbalanced data is that the cost of misclassification of the minority class (i.e. cost of false positives) is higher than of the majority class (i.e. cost of false negatives). Usually, the cost of the correct classification is equal to zero. For a given specific domain, the cost matrix can be provided by an expert. If such information is available, it indeed should be used and this methodology is natural to solve the given data mining problem (see Subsection 2.4.2). However, such information is rarely available. Thus the relevant setting for the cost matrix should be performed based on the available data during the learning stage, which is a difficult task (see e.g. [122, 138]).

There are three main approaches using the cost-sensitive methodology for the imbalanced learning problem. First, there are the methods which incorporate cost-sensitive functions to the standard ML algorithms in order to build cost-sensitive classifiers. Among them are cost-sensitive decision trees, cost-sensitive neural networks, cost-sensitive Bayesian classifiers, and cost-sensitive support vector machines. For example, for decision trees, the cost-sensitive function is used to choose the best condition to split the data and determine whether a subtree should be pruned (see e.g. [30, 57, 132]).

Second, there are the methods which use relevant weights for learning examples to redefine the distribution in order to improve classification performance (see e.g. [69]). For instance, boosting algorithms tend to generate distributions aiming to classify properly hard examples in the training data set (see e.g. [69]).

Third, there are methods based on Bayesian decision theory (see e.g. [45]).

It is worth noting that there exist some theoretical relationships between cost-sensitive learning and resampling techniques (see e.g. [57, 233], for other references see [38]). For example, a similar effect can be obtained using cost-sensitive learning and resampling technique.

2.5.4 One class learning

In case of the standard binary classification problem, imbalanced or not, classifier aims to discriminate instances of both classes. One-class learning are methods aiming to recognise instances only from the minority class. In this case, the training set contains mainly or only objects from that class. In consequence, the hypothesis construction of such classifiers naturally focuses on the minority class. For example, there exist one-class Support Vector Machines (see e.g. [143, 174]), one-class neural networks (see e.g. [111, 142]). The latter is based on the so-called autoassociator (or autoencoder). There exist also one-class rule-based learning algorithms (see e.g. [176], [236]). They learn only rules for the minority class (the majority class is not learnt at all).

It was reported in the literature that one-class learning is particularly useful for dealing with very imbalanced data sets and high dimensional feature space (see e.g. [174]).

2.5.5 Ensemble methods

Ensemble-based classifiers use a set of base learning algorithms. Each of them induces a classifier. Next, the obtained classifiers are combined to obtain a new classifier with (optimistically) better performance than each of these classifiers (see e.g. [139]). For example, ensemble of rule-based classifiers was successfully used for imbalanced data (see e.g. [25]).

2.6 Evaluation of learning algorithms

One of the important aspects of ML is a proper evaluation of the constructed systems, in particular the learning algorithms. This constitutes quite many topics. Many aspects concerning this issue are discussed in [109].

In [51], a taxonomy of possible statistical questions related to the evaluation of classifiers is presented. First, the fundamental question is whether we focus only on a single application domain or multiple domains. The answer to this question, in this thesis, is 'multiple domains' since we focus on inventing learning algorithms which could be used on a possible wide range of application domains.

To make the problem simpler, let us, for now, limit our considerations to two algorithms. Then the fundamental question one should answer is:

> *Given two learning algorithms A and B and data sets from several domains, which algorithm will produce more accurate classifiers when trained on examples from new domains?* [51]

Let us recall that we distinguish between a classifier and learning algorithm (see Section 2.1; see also [51]). In general, in the thesis, we present learning algorithms, which are compared with the other ones. However, as a step of such a comparison, one also needs a method to compare classifiers. If we focus on a comparison of classifiers, there arise other questions related to the taxonomy presented in [51]. Among others, we assume that the amount of supported data is limited (as it happens in many real-world learning tasks). Thus, one cannot use simple statistical methods

to compare two classifiers. Instead, one needs to use all the available data set as input. In consequence, we use a particular form of resampling called cross-validation. This raises more questions on how to use it properly to estimate the real value of the classifier's chosen performance measure with small bias.

It should be noted that the methodology used to evaluate the RIONA algorithm (see [81, 82]) is different from the one used in this thesis to evaluate the RIONIDA algorithm. Among others, this is due to the changing trends in the area of ML related to the issue of comparing learning algorithms.

Comparative studies usually include a new algorithm and several known methods. However, these studies should use very carefully their methods and their claims (see [181]). There are several steps important in the process of evaluation of learning algorithms:

1. choosing one or more performance measures relevant to the considered problem – the values of this measure for two classifiers (results of a learning algorithm for a given training data set) create the basis for their comparison (i.e. estimation which one is better or worse according to this measure),

2. selecting a method for estimation of the value of the chosen performance measure(s)[13] for a single data set (usually the cross-validation is used, but not always it is possible),

3. choosing a family of data sets on which estimation of the quality of given learning algorithms in terms of this(these) performance measure(s) should be done – these data sets are assumed to be representative for the real-life problems for which the considered classifiers will be applied (the aim is to construct classifiers with high quality for real-life problems or a specific family of such problems),

4. deciding which algorithm(s) from a given family is/are the best/comparable on the given data sets (usually this is done based on some statistical methods),

5. decision making based on the previous step indicating which learning algorithm from a given family is the best for a range of real-life data sets (or which algorithms are comparable).

Each of these steps is important and each presents its own difficulties. In the area of data mining, there are some generally accepted paths through these steps. However, each of these paths has its specific drawbacks. It should be emphasised that all of these steps could be seen as forming or influencing the logical/statistical inference concerning the comparison of two or more learning algorithms, potentially leading to some errors or bias. Thus even if we make every effort to be as formal as possible, this inference leads to some uncertainties and contains some gaps, errors, and weaknesses. One should bear this in mind when coming to the final conclusions. This also concerns the outcome of the experiments presented in this thesis (see Chapter 5).

[13] In analogous contexts, instead of 'value of performance measure' we often write 'performance measure', for short.

In the following subsections, all these steps are briefly described (see Subsections 2.6.1-2.6.5 for the steps 1-5, respectively). Also, brief drawbacks of the chosen paths are described.

2.6.1 Performance measures

The selection of relevant *performance measure*[14] is one of the key factors in assessing the classification performance and searching for the high-quality classifiers (see e.g. [139]). Finally, the relevant performance measure should be selected for a specific domain (see Subsection 2.4.2). In the absence of knowledge about it, a measure from the standard performance measures is usually used.

Confusion matrix

For evaluation of classifiers, a *confusion matrix* is often used (see e.g. [182]). A confusion matrix summarises the performance of classifier on a given test data. Any cell of the confusion matrix is identified by two indices. Every cell contains information on the number of objects belonging to the class indicated by the first index and classified to the class indicated by the second index.

Table 2.2 presents an example of a confusion matrix for a three-class classification task, with the classes d_1, d_2, and d_3. The first row of the matrix indicates that 11 objects belong to the class d_1 and that 7 are correctly classified as belonging to d_1, one misclassified as belonging to d_2, and tree misclassified as belonging to d_3.

In the thesis, such general confusion matrices are used for balanced data only.

		Predicted (Classified as)		
		class d_1	**class d_2**	**class d_3**
Actual (Really is)	**class d_1**	7	1	3
	class d_2	0	4	2
	class d_3	1	2	3

Table 2.2: An example of a three-class confusion matrix.

For balanced data, the *accuracy* rate is the most commonly used performance measure. In terms of confusion matrix, the *Accuracy* (*Accuracy* measure) is the sum of numbers in the diagonal divided by the number of all objects (sum of all numbers in the matrix)[15].

Performance measures used in the thesis for imbalanced data

As it was mentioned before, for imbalanced data we consider in the thesis only data sets with two decisions. For this case, the confusion matrix has a specific form

[14] In the literature, other names are also used, e.g. performance metric, assessment metric, evaluation metric, assessment measure.

[15] We write Accuracy in capital letter (analogously as F-measure and G-mean) when we refer directly to this definition or Equation 2.14.

presented in Table 2.3, and each cell in this matrix has its own name. The positive class relates to the minority class, and the negative class relates to the majority class.

		Predicted (Classified as)	
		Positive	**Negative**
Actual (Really is)	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 2.3: Confusion matrix for a two-class problem.

In this case, the most common performance measure, *Accuracy* (*Accuracy* measure), is defined as follows.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{2.14}$$

Generally, *Accuracy* is the probability that for any test example the classification is correct (see Subsection 4.3.4 for the use of such definition). This measure is not relevant for imbalanced data sets, since it does not distinguish between the number of correctly classified examples from different classes.

The acceptable performance measures for imbalanced data are usually composed out of the following sub-measures (see e.g. [102]):

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.15}$$

$$Specificity = \frac{TN}{FP + TN} \tag{2.16}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.17}$$

Sensitivity is the conditional probability that the classification is correct given the actual positive class. *Specificity*, the complement measure to Sensitivity, is the conditional probability that the classification is correct given the actual negative class. *Precision* is the conditional probability that the classification is correct given the classifier predicts positive class.

Other names for these measures used in the literature are given in Table 2.4.

Now, we present important performance measures used in the thesis.

Widely used performance measure for imbalanced data is *F-measure* (see e.g. [13, 18, 49, 117, 238]), which is the harmonic mean of Precision and *Recall*, i.e. Precision and Sensitivity:

$$F\text{-}measure = 2 \cdot \frac{1}{\frac{1}{Sensitivity} + \frac{1}{Precision}} = 2 \cdot \frac{Precision \cdot Sensitivity}{Precision + Sensitivity} \tag{2.18}$$

Name usually used in the thesis	Other names used in the literature
Sensitivity	True Positive Rate, Accuracy for Positive Class, Recall
Specificity	True Negative Rate, Accuracy for Negative Class
Precision	Positive Predictive Value

Table 2.4: Different names for given measures.

The last equality holds under the assumption that both Sensitivity and Precision are not equal to zero, which is equivalent to the assumption that True Positive value (TP) is not equal to zero.

The presented formula is a specific case of more general one F_β-*measure*, where by the parameter β the different importance of Precision and Sensitivity can be set. The presented definition of F-measure corresponds to the case when $\beta = 1$, i.e. to the case with the equal importance of Precision and Sensitivity. In the thesis, only this case, presented in Equation 2.18, is used. This performance measure and its properties are widely discussed in the literature (see e.g. [46, 65, 98, 133, 163]). There are also other, more sophisticated performance measures based on the F_β-measure (see e.g. [144]).

Another widely used performance measure for imbalanced data is *G-mean* (see e.g. [13, 18, 58, 125, 194, 238]), which is the geometric mean of Sensitivity and Specificity:

$$G\text{-}mean = \sqrt{Sensitivity \cdot Specificity} \tag{2.19}$$

By substituting Sensitivity and Precision in Equation 2.18 and using Equations 2.15, 2.17 one can express F-measure in terms of True Positives, False Negatives and False Positives. After a few simple calculations, assuming that $TP + FP \neq 0$ and $TP + FN \neq 0$, we obtain the following equation:

$$F\text{-}measure = \frac{2 \cdot TP}{2 \cdot TP + FN + FP} \tag{2.20}$$

Condition $TP+FP \neq 0$ does not hold (e.g. Precision is undefined) when classifier makes no positive predictions. Condition $TP+FN \neq 0$ does not hold (and Sensitivity is undefined) when there are no positives in the considered set.

For practical use (and to be precise), one should specify either how one treats the exceptional situations or define the used measures for them. We assume (which is normally true, in particular, true for our experimental setup; see Chapter 5) that for all considered situations, the number of both majority and minority examples is nonzero (i.e. $TP+FN \neq 0$ and $FP+TN \neq 0$). Then, Sensitivity and Specificity (and thus G-mean) are well-defined. Under the mentioned assumption, also, Equation 2.20 is well-defined. We use this definition as an extension of F-measure such that it is well-defined (namely, zero) in all other exceptional situations mentioned earlier[16].

[16]It is equivalent to use the original definition (see Equation 2.18) of F-measure and return value zero for all undefined situations.

Other performance measures for imbalanced data

Receiver Operating Characteristics (*ROC*) *Analysis* is used to distinguish performance between classes (i.e. Sensitivity and Specificity) of binary classifiers for different decision thresholds (see e.g. [182]). In practice, the *ROC curve* is used. It is a graphical plot that visualises the relation between Sensitivity (True Positive Rate) and 1 − Specificity (the False Positive Rate) for a classifier under varying decision thresholds. It should be noted that the G-mean measure relates to a point on the ROC curve which represents the balance between Sensitivity and Specificity in attempt to maximise both the components (see e.g. [124]).

Analogously, to ROC Analysis, *Precision-Recall Analysis* is also used (see [66]). It should be noted that F-measure relates to a point on the Precision-Recall curve which represents the balance between Precision and Sensitivity in attempt to maximise both components.

The widely used performance measure is *Area Under the ROC Curve* (*AUC*) which is a summary statistic of ROC Analysis (see e.g. [109]). However, the AUC measure has also serious drawbacks (see e.g. [97, 172]). In the literature, many other performance measures relevant for imbalanced data are proposed (see e.g. [31, 91, 97, 102, 109]).

2.6.2 Estimation of the chosen performance measure

We assume here that the performance measure is fixed. We also assume here that the data set is given, but the number of examples in this set is small relative to all possible examples. The important issue now is to estimate the value of the chosen performance measure for a given learning algorithm and the considered domain using the given data set. We would like to obtain an estimate of the considered performance measure as unbiased as possible with the property of reproducibility (see e.g. [171]). This is not an easy task (see e.g. [29, 109, 171]). The most popular estimation technique in ML is the k-fold cross-validation. It divides a given data set D into k disjoint subsets $D^1, \ldots, D^k$ (called folds) of roughly equal sizes. In i-th iteration, the learning algorithm is trained on $D \setminus D^i$ set and tested on the D^i set. Thus in each iteration, we obtain a separate number estimating the performance of the classifier. Usually, the average of those numbers is used as an overall estimation of the selected measure for the learning algorithm. A typical choice of k is 10, which is recommended in [120] and also used in the thesis.

In the standard cross-validation, the distribution of classes is not taken into account. However, for imbalanced data the minority class may be under-represented or even absent in some folds used as the test set. It may result in the biased estimation. Thus, for imbalanced data the stratified cross-validation is used. It takes care that in each fold D^i the distribution of classes is roughly the same as in the original set. Testing procedure in the thesis (for imbalanced data) is always done with the stratified cross-validation. It should be noted that the criticism of the stratified cross-validation as well as a new, more sophisticated estimation methods for imbalanced data are presented in the literature (see e.g. [140]).

Let us return to the issue of how to compute one overall estimation *score* having system outputs for each of the k folds. This issue relates to the micro- or

macro-average style. In the case of Accuracy, the usual procedure of averaging the results of Accuracy for each of the k folds is satisfactory. Under the assumption that all folds are exactly of the same size, the joining confusion matrices of all folds and computation of Accuracy for such joint matrix gives exactly the same result[17]. This is not true for such measures as F-measure or G-mean. For the sake of simplicity, let us assume that F-measure was chosen as the performance measure. The macro-average style means counting F-measure for each fold and finally counting the average of these numbers. The micro-average style means that all True Positives, False Negatives, False Positives, and True Negatives are summed over all folds. With these counts, the F-score is computed. In other words, F-measure in the micro-average style is counted from the joint confusion matrix (with coefficients equal to the sum of coefficients from each fold).

It was shown in [67] that simple averaging of separate results, i.e. the macro-average style can give the biased estimation. However, the micro-average style gives less bias. Authors also inform about other possible ways to estimate F-measure using the cross-validation scheme. This is related to the issue of how special cases are treated, which was discussed in the previous subsection. Analogously, estimation of G-mean value may be influenced by choosing the method of computation of the final performance result.

During experiments, we have found that choosing the way of G-mean computation (i.e. using the micro- or macro-averaging) influences not only the bias but, potentially, also the results of the global comparison of classifiers. We found the situations when the order of classifier performance relative to G-mean can be reversed by changing the method of averaging (see Appendix B). This suggests that one should be very careful in choosing the way how the cross-validation results are aggregated. Moreover, including reports from experiments showing how G-mean or F-measure is computed is important (in some papers, such information is not included).

In the thesis, we use the micro-average style of computation for all performance measures, i.e. F-measure and G-mean.

It should be noted that also the repeated cross-validation is often used. The estimated values in the consequent experiments are averaged. However, one should be conscious of the problems related to repeated cross-validation and that overusing it may lead to false conclusions (see e.g. [202]). In our experiments, we use 10 times repeated 10-fold stratified cross-validation.

2.6.3 Selection of data sets for evaluation

This step is strongly related to step 5 (see Subsection 2.6.5).

In general, no learning algorithm is the best for all possible problems. Formally this fact is well known as the so-called 'no free lunch theorem' (see [223]). However, learning algorithms are to be used not for the set of all possible mathematical concepts but for real-world domains (see [203, p. 721]).

In practice, when we need to compare a few learning algorithms, one can choose several data sets related to some real-world domains and comparison over these sets can be used for comparison in which we are interested. Often the UCI repository

[17] With different fold sizes using weights related to the sizes of folds would give equal results.

is used for this purpose. In this thesis, we also mainly choose data sets from this repository as representatives for the classification problem we want to solve.

When choosing data sets for comparisons, one should consider the fact that different data sets may appear there: with numerical attributes, with symbolic attributes or with mixed attributes. If one decides to use the cross-validation (step 2), then data sets which are not appropriate for using cross-validation cannot be selected for analysis.

As our aim is to build learning algorithms of the high quality for imbalanced learning problem, we want to select imbalanced data sets for analysis. Moreover, in this case, one should consider data sets with different levels of difficulty (see Subsections 2.4.2, 2.4.3).

2.6.4 Statistical tests

This is the fourth step (see the beginning of Section 2.6). We assume here that the representative (i) data sets (for real-world problems), (ii) state-of-the-art algorithms to compare with are chosen. Also, we assume here that the relevant performance measure was selected and we have a good estimation of the value of this measure for any pair consisting of algorithm and data set. Now, the important issue is to decide which of the algorithms is the best for the chosen data sets.

The observed differences among the performance of algorithms (in terms of chosen performance measure) may come from real differences between algorithms or due to randomness, e.g. from the specific use of data set, from the specific splitting used in the cross-validation (both random variation of test data, and random variation of training data), internal randomness in the learning algorithm or noise in data set. To check whether the differences between the performance of algorithms are due to the real differences, some *statistical tests* are used.

For our purpose, we need a test tool making it possible to compare multiple algorithms on multiple data sets. It should be noted that this is a much more complex experimental design than in case of comparing algorithms on a single data set or comparing only two algorithms on several data sets.

Mainly this is because many comparisons are done and *family-wise-error* (the probability of making at least one type I error in any of the comparisons) should be controlled. To make our task simpler, we can also use the fact that we are generally focused on comparing one algorithm (e.g. RIONIDA presented in the thesis) with other state-of-the-art algorithms.

The most popular statistical methodology in the ML community nowadays can be summarised as follows. First, we apply a joint test to check whether at least one of the algorithms performs better than the other ones. The null hypothesis is that all algorithms perform equally well. Second, if the null hypothesis is rejected, i.e. a significant difference is detected, then we can proceed with a post-hoc test (to check between which pairs of algorithms there are actually significant statistical differences). The null hypothesis in the second step is that two algorithms chosen for comparison perform equally well (and such hypothesis is checked for many pairs of algorithms).

In experiments used in the thesis, we use the significance level $\alpha = 0.05$. When testing a hypothesis, one can be more informative than simply reporting 'reject'

or 'accept' by using so-called *p-value* (see e.g. [80, 240]), a well-known concept in statistics. In statistical hypothesis testing, the *p-value* is the probability of obtaining a result at least as extreme as the one that was actually observed, assuming that the null hypothesis is true. In practice, the null hypothesis is rejected when the p-value of the corresponding test is less than α. However, the smaller the p-value, the stronger is the evidence to reject the null hypothesis (see e.g. [240]).

We use the *Friedman statistical test* (see [70]; see also [47]) for the first step. It is a non-parametric counterpart of the well-known ANOVA test. The Friedman test *ranks* the algorithms, i.e. for each data set, the algorithms are sorted according to the selected performance measure, and numbers from 1 to the number of algorithms are assigned. In the case of ties, the average ranks are assigned. Then average ranks over data sets are calculated, and the Friedman statistic is computed (for details see e.g. [47]). This test takes into account the variations in the ranks of algorithms. Let K, N be the total number of algorithms and data sets used in the comparison, respectively. Under the null hypothesis, which states that all compared algorithms perform equally well, the Friedman statistic follows the Chi-square distribution with $df = K - 1$ degrees of freedom, when N and K are not too small (e.g. $N > 10$ and $K > 5$)[18].

One can apply a pair-wise test with the corresponding post-hoc correction for multiple comparisons (see e.g. [34, 73]) as a test for the second step. This is used when all learning algorithms are compared against each other. One can use *Nemenyi statistical test*[19] (see [156]; see also [109]) for this. Although the Nemenyi test is a very conservative procedure (has a high type II error), we sometimes use it because its results can be represented in a critical difference diagram interpreted as follows. The closer to the left (lower average ranks), the better algorithm is. Also, the groups of algorithms that are not significantly different are connected by a horizontal line (see the example in Figure 5.6 on page 167).

When all learning algorithms are compared with a control one (in our experiments it is the learning algorithm presented in the thesis, i.e. RIONIDA), one can use other post-hoc procedures with higher power than the Nemenyi test. First, these procedures compute statistics for comparing any learning algorithm with the control one. Any comparison is associated with the null hypothesis that the control learning algorithm performs equally well as the compared one. For each comparison p-value is computed. Next, these procedures report adjusted p-values (APVs) which take into account that multiple tests are conducted (to control family-wise-error rate). The simplest one is Bonferroni correction (see e.g. [47]). It adjusts p-values by multiplying them by the number of comparisons, i.e. $K-1$. Among more complex tests is the *Finner statistical test* (see [64], [75]). Since it is reported in the literature as the test with high power (see e.g. [75, 198]), we decided to use it in our experimental design.

It should be noted that some researchers criticise such an approach using null hypothesis significance testing and suggest using the Bayesian approach instead (see [22, 208]).

[18] Since in the thesis we always use $N = 20$ and $K = 10$, this assumption is satisfied.

[19] It is similar to Tukey test for ANOVA.

2.6.5 Selecting the best learning algorithm for real-life data sets

Based on the previous steps, we decide which algorithm from a given family is the best for a range of real-life data sets (or which algorithms are comparable). Let us for a while assume that all the selections and evaluations done in steps 1-4 (see Subsections 2.6.1-2.6.4) were perfect. Let us assume that we found that our new algorithm is statistically significantly better than the other ones (decided in step 4, see Subsection 2.6.4). Idealistically, one could conclude that this algorithm is the best (or at least not worse) for a specific subset of real-life classification tasks (in our case classification tasks with imbalanced data).

However, we would like to briefly recall some problems related to such inference. First, if one picks up data sets from a population (usually it is the UCI repository) to carry out an experiment, any inferences one makes can only be applied to the original population itself. Thus, it is not valid to make general statements about other data sets (see [181]). Second, if many researchers make a statistical test on the same small repository of data sets, then the chance of making false conclusions is growing up (see e.g. [181]). The problem can be even more serious when researchers tune the parameters of their algorithms (see [181]).

2.6.6 Conclusions about the evaluation of learning algorithms

First, we would like to underline that the above-described inference process is not easy and sometimes may lead to false conclusions even if only one of the presented steps is not prepared perfectly. However, suppose some learning algorithm turns out to be the best (according to the described inference) for some real-world problems; this can be regarded as an argument suggesting that such an algorithm should be taken into account when other real-world domains are given.

Second, let us summarise that in the case of imbalanced data a few very important aspects should be taken into account:

- performance measure should be different from Accuracy, for example, F-measure or G-mean should be taken,
- the stratified cross-validation should be used,
- during cross-validation the micro-average should be used as a method of collecting results from subsequent trials,
- proper data sets should be chosen reflecting the difficulty of imbalanced data.

2.7 Summary of the chapter

This chapter recalls some concepts known from the literature, the most important of which for the thesis are:

- the problem of supervised learning; classifier and learning algorithm, with emphasis on differences between them;

- metric and pseudometric – important concepts for instance-based learning; also used in the thesis for grouping of values of attributes;
- rule-based methods and the set of all rules that are maximally general and consistent with a training set; the *Strength* measure for conflict resolution – its modified version is used in the next chapter;
- instance-based learning;
- a specific lazy rule learning for symbolic attributes only – generalised in the next chapters for the rules commonly used in the thesis;
- imbalanced data – difficulties in their analysis and different algorithmic approaches, the most important of which is the algorithm-level approach applied in the thesis;
- confusion matrix and the most important performance measures used in the thesis, namely Accuracy, G-mean, and F-measure;
- cross-validation estimation technique, and the method (used in the thesis) of collecting its results, namely micro-average;
- Friedman statistical test and Finner test (and, sporadically used, Nemenyi test).

In this chapter, some concepts are also defined for the thesis, of which we would like to emphasise:

- elementary condition of rules used for grouping values of an attribute;
- the set of general rules out of three kinds of decision rules (according to admissible elementary conditions) presented in the chapter; this set will be used to show the relationship between rules commonly used in the thesis and more general lazy learning approach defined in the next chapter;
- pseudometric decision system (based on decision system concept, known from the literature and also introduced in this chapter);
- general aggregated pseudometric and the default aggregated pseudometric used in the thesis for measuring the distance between objects, namely City And Simplified Value Difference pseudoMetric (CSVDM).

Among others, we presented, known from the literature, the equivalence of lazy rule learning for symbolic attributes only with the simple rule-based approach. This result will be generalised in the next chapter for more general rules commonly used in the thesis.

Chapter 3

RIONA

RIONA is the acronym of Rule Induction with Optimal Neighbourhood Algorithm. This algorithm is designed for balanced data sets. It is constructed to maximise Accuracy performance measure.

The following section introduces the main ideas behind the RIONA algorithm. Section 3.2 introduces an extension of lazy rule learning for numerical attributes, and its generalisation for symbolic attributes. Section 3.3 describes the testing phase of the RIONA algorithm and its time complexity. Moreover, it shows relationships of RIONA with instance- and rule-based classifiers. Section 3.4 describes the training phase of the RIONA algorithm and its time complexity. Section 3.5 presents a summary of the experimental properties of RIONA. Section 3.6 briefly outlines some possible extensions of the basic version of RIONA, described in this chapter. Section 3.7 briefly introduces the idea how RIONA can be extended for imbalanced data. Finally, Section 3.8 concludes this chapter.

The RIONA algorithm has three parts: initialisation, training and testing. Some comments on the formal structure of the whole RIONA algorithm can be found in Subsection 3.4.3.

Most of the work presented in this chapter, especially the RIONA algorithm's development, was carried out in collaboration with Wojna (see Section 1.6). Independently, the author of the thesis: (i) developed a new form of presentation of foundations leading to RIONA, (ii) formulated and proved facts better explaining the relationships of RIONA with rule-based classifiers, and (iii) proposed a user-friendly explanation method of the decisions returned by the classifiers obtained from RIONA.

3.1 Main ideas behind the RIONA algorithm

The algorithm was developed using some general ideas and at the same time some specific ones (in particular, defined by default parameter settings used in the main experiments) which are shortly described below.

RIONA is based on the LAZY algorithm (see Algorithm 2) presented in Subsection 2.3.2. In the thesis, we extend this algorithm for numerical attributes and for symbolic attributes more general conditions than that of the LAZY algorithm (see Section 3.2) are taken.

The decision is predicted using the support set restricted to a neighbourhood of

the test case (see Section 3.3) rather than the whole support set of all rules (calculated on the training set) covering the case.

In the realisation of these two ideas concerning generalised rules and instance-based learning, we use pseudometrics (see comments concerning pseudometrics in the thesis in Subsection 2.2.3). Pseudometrics are used for two reasons.

First, pseudometrics are used for grouping attribute values in the construction of generalised rules. For symbolic attributes, it is assumed that pseudometrics are provided relative to the given training set. As a default, SVDM pseudometrics corresponding to symbolic attributes are used (see Subsection 2.2.2). They are calculated from the training sets during the learning phase.

Second, because classification by RIONA is based on neighbourhoods, we also use pseudometrics over objects. The neighbourhoods are constructed by searching for nearest neighbours for the test objects (see Subsection 2.3.3, and Section 3.3). As it was mentioned in Subsection 2.2.3, we assume by default that the specific aggregated pseudometric for objects are constructed from pseudometrics for attributes (see Equation 2.1). Thus, by default, for measuring the distance between objects, pseudometric CSVDM is used.

An important feature of RIONA is that the optimal neighbourhood can be estimated efficiently by using dynamic programming. Moreover, the performed experiments show that the searching space for this neighbourhood estimation can be extremely bounded without losing the classification quality (see Section 3.4).

Some relationships of the RIONA classifier to both instance-based and rule-based classifiers are also presented in the thesis (see Subsections 3.2.2, 3.3.4 and 3.3.5). Moreover, a very interesting observation is made about the RIONA classifier concerning the possibility of representing the constructed optimal neighbourhood of the classifier by a rule set, with rules easily understandable by a human (see Subsection 3.3.5).

The empirical results indicate that the Accuracy of the constructed RIONA algorithm is comparable to the well-known systems (see Subsection 3.5.1).

3.2 Extension and generalisation of lazy rule learning

In this section, we present an extension and generalisation of the LAZY algorithm (see Algorithm 2) presented in Subsection 2.3.2. We extend this algorithm to the case of numerical attributes and we use more general conditions for symbolic attributes.

We present our final idea, in three steps. The first step is described in Subsection 2.3.2. In two remaining steps, presented in this section, we use a generalisation of rules from the previous step (see Subsections 3.2.1, 3.2.2).

Thus, in this section together with the first step from Subsection 2.3.2, we define three types of local rules: simple local decision rule, *combined local decision rule* and *generalised local decision rule* (for short, s-rule, c-rule, and g-rule, respectively), denoted by $s\text{-}rule(tst, trn)$, $c\text{-}rule(tst, trn)$, $g\text{-}rule\left(tst, trn, \{\varrho_a\}_{a \in A_{sym}}\right)$ (or simply $g\text{-}rule\left(tst, trn\right)$), respectively, where trn is the training object, tst is the test object

and ϱ_a for $a \in A_{sym}$ are pseudometrics defined by pseudometric decision system. The introduced names correspond to the sets composed out of simple rules, combined rules and general rules, denoted by $SimRules$, $CombRules$, $GenRules$, respectively (see Subsection 2.3.1). In Subsection 2.3.2, an important relation between any s-rule and the set of maximally general consistent rules $MaxRules(SimRules, trnSet)$ is presented. In this section, we show analogous important relations between any c-rule or g-rule with their corresponding sets of maximally general consistent rules (denoted by $MaxRules(CombRules, trnSet)$ and $MaxRules(GenRules, trnSet)$, respectively).

3.2.1 Extension of lazy rule learning for numerical attributes

Throughout this section, we assume that a decision system $\mathbb{D} = (\mathbf{X}, A, d)$ and a training set $trnSet \subseteq \mathbf{X}$ are given.

In the second step, we define an extension of the local decision rule to the case of both symbolic and numerical attributes.

Definition 3.1. *For any test object tst and any training object trn, we define the* combined local decision rule *(for short* c-rule*), denoted by c-rule(tst, trn), with the decision $d(trn)$ and the following conditions T_a for each attribute $a \in A$:*

$$T_a = \begin{cases} a \in [min_a, max_a] & \text{if } a \text{ is numerical} \\ t_a & \text{if } a \text{ is symbolic,} \end{cases}$$

where t_a is defined as in Definition 2.13, $min_a = min(a(tst), a(trn))$, $max_a = max(a(tst), a(trn))$.

For numerical attributes (linearly ordered), conditions are represented in the form $a \in [min_a, max_a]$. The interval's endpoints are determined by the attribute values of the examples tst and trn used to form the rule.

In Figure 3.1, an exemplary area of satisfiability of the rule $c\text{-}rule(tst, trn)$ for objects tst, trn is illustrated in the case of a data set with two numerical attributes. The satisfiability area of the c-rule is represented by a rectangle spanned over the points with coordinates determined by attribute values of each of the examples tst, trn.

It should be noted that by defining c-rule in such a way we obtain an analogous relationship of the set $MaxRules(CombRules, trnSet)$ and $c\text{-}rule(tst, trn)$ to the relation between $MaxRules(SimRules, trnSet)$ and $s\text{-}rule(tst, trn)$.

Lemma 3.1. *Any rule $r \in MaxRules(CombRules, trnSet)$ covering the given test object tst and training object trn is implied by the rule c-rule(tst, trn).*

Proof. Since r covers tst and trn and is consistent (with $trnSet$), we have the following. For each attribute $a \in A$, $t_a(r)(trn)$ is satisfied and $t_a(r)(tst)$ is satisfied, i.e. $trn \in [[t_a(r)]]_{\mathbb{D}}$ and $tst \in [[t_a(r)]]_{\mathbb{D}}$.

For rule r such that the elementary condition $t_a(r)$ is of the form $a \in V$ let us define $V_a(r) = ||V||_{\mathbb{D}}$. We will show that for all $a \in A$ the implication $t_a(c\text{-}rule(tst, trn)) \Rightarrow t_a(r)$ holds, i.e. $V_a(c\text{-}rule(tst, trn)) \subseteq V_a(r)$.

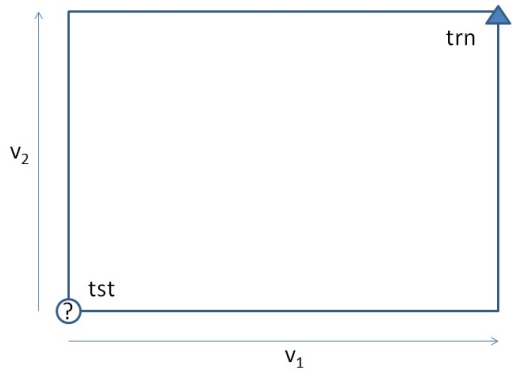


Figure 3.1: Illustration of the area of satisfiability of the rule *c-rule*(tst, trn) defined by two objects *tst* and *trn* for a data set with two numerical attributes. The difference between attribute values of *tst* and *trn* on the first and the second attribute is v_1 and v_2, respectively.

First, let us consider the case when a is symbolic. If $t_a(r)$ is a trivial condition, i.e. $t_a(r)$ is of the form $a \in V_a$, then the implication obviously holds (trivial condition is implied by any condition, because for any elementary condition $a \in V$ for attribute a, $||V||_{\mathbb{D}} \subseteq ||V_a||_{\mathbb{D}}$). Let us consider the case when $t_a(r)$ is of the form $a = v$ for some $v \in V_a$. Then, because $t_a(r)(trn)$ and $t_a(r)(tst)$ are satisfied, then $trn \in [[a = v]]_{\mathbb{D}}$ and $tst \in [[a = v]]_{\mathbb{D}}$, i.e. $a(trn) \in \{v\}$ and $a(tst) \in \{v\}$, thus $v = a(trn) = a(tst)$. It means that $t_a(r) = t_a(c\text{-}rule(tst, trn))$ (see Definition 3.1 and Definition 2.13).

Second, let us consider the case when a is numerical. Thus $t_a(r)$ is of the form $a \in I$, where I is the interval corresponding to the numerical attribute a of rule r. Because $t_a(r)(trn)$ is satisfied, i.e. $trn \in [[t_a(r)]]_{\mathbb{D}}$ and $t_a(r)(tst)$ is satisfied, i.e. $tst \in [[t_a(r)]]_{\mathbb{D}}$ and by definition $[[a \in I]]_{\mathbb{D}} = \{x \in \mathbf{X} : a(x) \in ||I||_{\mathbb{D}}\}$ we have $a(trn) \in ||I||_{\mathbb{D}}$ and $a(tst) \in ||I||_{\mathbb{D}}$, thus $\{a(trn), a(tst)\} \subseteq ||I||_{\mathbb{D}}$. Thus, all points between $a(trn)$ and $a(tst)$ are also in $||I||_{\mathbb{D}}$. In consequence, $[min_a, max_a] \subseteq ||I||_{\mathbb{D}}$, where $min_a = min(a(tst), a(trn))$, $max_a = max(a(tst), a(trn))$. Thus, $V_a(c\text{-}rule(tst, trn)) \subseteq V_a(r)$ (see Definition 3.1). □

Let us note that in the following proofs, we omit some formal details such as used in the above proof.

Theorem 3.2. *The rule c-rule*(tst, trn) *for the test object tst and the training object trn is consistent with the training set trnSet if and only if there exists a rule* $r \in MaxRules(CombRules, trnSet)$ *covering objects tst and trn.*

Proof. First, we show that if the rule *c-rule*(tst, trn) is consistent with the training set *trnSet*, it can be extended to a rule belonging to $MaxRules(CombRules, trnSet)$. We define such a rule inductively. Rule $r_0 = c\text{-}rule(tst, trn)$ is in *CombRules* and is consistent with *trnSet* by assumption. The induction step is as follows. To define each next rule r_i, for $i = 1, 2, \ldots, m$, where m is the number of attributes, we assume that rule r_{i-1} is consistent with *trnSet* and conditions $t_j(r_{i-1})$ for all $j = 1, 2, \ldots, i-1$ are maximally general, i.e. if we replace any condition t_j with a more general t (i.e. $t_j \Rightarrow t$) preserving consistency, then $t_j = t$.

In the i-th induction step we define the condition $t_i(r_i)$ as the maximal generalisation of the condition $t_i(r_{i-1}) = t_i(r_0) = t_i(c\text{-}rule(tst, trn))$ preserving

consistency with $trnSet$. All others conditions and the decision of the rule are defined as in the previous induction step, i.e. $t_j(r_i) = t_j(r_{i-1})$ for $j \neq i$; $d(r_i) = d(r_{i-1})$. In other words, in i-th induction step we simply maximally generalise condition for attribute a_i.

First, let us consider the case when a_i is symbolic. If $t_i(r_{i-1})$ is the trivial condition, then we define $r_i = r_{i-1}$. If $t_i(r_{i-1})$ is non-trivial, we replace it with the trivial condition if such replacement keeps the consistency of the rule; otherwise, we keep $r_i = r_{i-1}$.

Now, let us consider the case when a_i is numerical. Thus $t_i(r_{i-1})$ is of the form $a_i \in [min, max]$. We define by $rule_i(r, t)$ the rule r with the replacement of i-th condition in it by condition t. Let us consider the set of training examples which potentially may violate the consistency of the rule under the maximal possible extension of the condition $t_i(r_{i-1})$, i.e. the set $Inc = \{trn \in trnSet : d(trn) \neq d(r_0) \wedge rule_i(r_{i-1}, a_i = *) \text{ covers } trn\}$. Let us define $a(Inc) = \{a(trn) : trn \in Inc\}$. From the induction assumption, r_{i-1} is consistent with $Inc \subseteq trnSet$. Thus we have $a(Inc) \cap [min, max] = \emptyset$. Let us define $newmax = min\{v \in a(Inc) : v > max\}$. Let us note that this minimum exists because the Inc set and therefore also $a(Inc)$ are finite sets. If the set $\{v \in a(Inc) : v > max\}$ is empty we define $newmax = u_{a_i}$ (i.e. maximal possible extension of the right end of the interval). Analogously, let us define $newmin = max\{v \in a(Inc) : v < min\}$. If the set $\{v \in a(Inc) : v < min\}$ is empty we define $newmin = l_{a_i}$ (i.e. maximal possible extension of the left end of the interval). Finally, we define $t_i(r_i)$ as the condition $a \in (newmin, newmax)$. From the definition, r_i is consistent with $trnSet$. It is also maximal because maximally extended ends of the interval (i.e. l_{a_i} or u_{a_i}) cannot be extended and other ends of the interval even if extended by one point to a closed interval will cause inconsistency.

We prove now that all other conditions $t_j(r_i)$ for $j < i$ are still maximal. Let us assume that for some $j < i$ the condition $t_j(r_i)$ could be extended to t with preserving consistency, i.e. $rule_j(r_i, t)$ is consistent. We also have that $rule_j(r_i, t)$ is more general rule than $rule_j(r_{i-1}, t)$ (i-th condition is more general), i.e. $rule_j(r_{i-1}, t) \Rightarrow rule_j(r_i, t)$. Therefore $rule_j(r_{i-1}, t)$ is consistent with $trnSet$. From the induction assumption $t = t_j(r_{i-1})$. Because $t_j(r_{i-1}) = t_j(r_i)$ for $(j < i)$, then $t = t_j(r_i)$. It means that $t_j(r_i)$ is maximally general.

By induction, the last rule r_m is consistent with $trnSet$ and maximally general.

Second, we show that if the rule $c\text{-}rule(tst, trn)$ is inconsistent with the training set $trnSet$, then there is no rule in $MaxRules(CombRules, trnSet)$ covering tst and trn. In fact, from Lemma 3.1 we have that each rule $r \in MaxRules(CombRules, trnSet)$ covering objects tst and trn is implied by $c\text{-}rule(tst, trn)$. Thus inconsistency of the rule $c\text{-}rule(tst, trn)$ implies inconsistency of all rules $r \in MaxRules(CombRules, trnSet)$ covering tst and trn.

□

3.2.2 Generalisation of lazy rule learning for symbolic attributes

From now on, we assume that a pseudometric decision system $\mathbb{D} = (\mathbf{X}, A, d, \{\varrho_a\}_{a \in A})$ is given. As before, we also assume that a training set $trnSet \subseteq \mathbf{X}$ is given.

In the previous Definitions 2.13 and 3.1, the trivial condition for symbolic attributes is used (when attribute values of the test and training examples differ). This condition represents the grouping of all possible values of an attribute and is satisfied by any object. However, we noticed that a proper subset of all attribute values can be more relevant for the classification. Grouping of values can be done using a given pseudometric for the attribute. Hence, as the third and final step, we propose the following generalisation of Definition 3.1, which additionally leads to a grouping of values for symbolic attributes:

Definition 3.2. *For any test object tst and any training object trn, we define the* generalised local decision rule *(for short* g-rule*), denoted by g-rule* $\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}\right)$ *or simply g-rule* (tst, trn) *(if parameters* $\{\varrho_a\}_{a\in A_{sym}}$ *are clear from the context or irrelevant due to generality of considerations), with the decision* $d(trn)$ *and the following conditions* t_a *for each attribute* a*:*

$$t_a = \begin{cases} a \in [min_a, max_a] & \text{if } a \text{ is numerical} \\ a \in B\left(a(tst), r_a\right) & \text{if } a \text{ is symbolic,} \end{cases}$$

where $min_a = min(a(tst), a(trn))$*,* $max_a = max(a(tst), a(trn))$*, the radius* $r_a = \varrho_a(a(tst), a(trn))$*, and* $B(c, R)$ *is a closed pseudometric ball of radius* R *centred at point* c *defined by the pseudometric* ϱ_a*.*

Please note that in this definition, we only use pseudometrics for symbolic attributes ($a \in A_{sym}$). However, in this definition (and in the previous one for the c-rule), for the numerical attributes, the natural order between its values and the natural Euclidean metric are assumed indirectly[1]. It should be noted that to obtain Proposition 3.6 we assume that pseudometrics for numerical attributes are weighted Euclidean metrics.

For numerical attributes (linearly ordered), conditions are the same as in c-rule, e.g. in the form of inclusion in closed interval. Symbolic attributes (non-ordered) are treated differently. For any such an attribute, a pseudometric defining distances among the attribute's values is required to be defined. The condition represents the specific group of values defined by a ball in the equation for t_a.

It is easy to check that if ϱ_a in Definition 3.2 is the discrete metric (see Subsection 2.2.2), then the conditions for symbolic attributes are equivalent to the condition used in Definition 2.13, i.e. when $a(tst) = a(trn)$, then the condition is $a = a(trn)$, otherwise the condition is the trivial condition. Thus, Definition 3.2 is a generalisation of Definitions 2.13 and 3.1 for all symbolic attributes and mixed attributes, respectively.

The conditions are chosen in such a way, that both the training and the test example satisfy the rule and the conditions are maximally specific. It means that

[1] One can interpret this interval as the result of grouping values into a ball $B(\frac{max_a+min_a}{2}, \frac{max_a-min_a}{2})$ with the Euclidean metric. If one would like to group values according to general pseudometrics for numerical attributes, it could be done analogously to symbolic attributes, i.e. $a \in B\left(a(tst), r_a\right)$, where $r_a = \varrho_a(a(tst), a(trn))$. Then, for the Euclidean metric we obtain, e.g. the interval $[2 \cdot a(tst) - a(trn), a(trn)]$ assuming that $a(tst) < a(trn)$. Thus, we would obtain the interval twice as large as the one used in the given definition. Such grouping interval seems not natural for real-valued attributes. This is the reason why numerical attributes are treated separately.

making the interval smaller for a numerical attribute or making the radius smaller for a symbolic attribute will cause the example trn not to satisfy the rule.

As an example let us consider again the data set presented in Table 2.1. Now, we calculate the rules *g-rule* (tst, trn_1) and *g-rule* (tst, trn_2). First, we calculate the closed balls used in the rules using calculated distances between values of attribute *BloodGroup* for pseudometric SVDM made in Subsection 2.2.2:

$$B(a_{BG}(tst), \varrho_{BG}(a_{BG}(tst), a_{BG}(trn_1))) = B(A, \varrho_{BG}(A, A)) = B(A, 0) = \{A\}$$

$$B(a_{BG}(tst), \varrho_{BG}(a_{BG}(tst), a_{BG}(trn_2))) = B(A, \varrho_{BG}(A, AB)) = B(A, \frac{4}{3}) = \{A, B, AB\}.$$

Then for the training object trn_1, the *g-rule* (tst, trn_1) which is equal to

$$\text{if } (Age \in [35, 50] \wedge Weight \in [72, 90] \wedge BG \in \{A\}) \text{ then } Diagnosis = Sick$$

is consistent because no other object from the training set satisfies the premise of this rule. But for the training object trn_2 the *g-rule* (tst, trn_2) which is equal to

$$\text{if } (Age \in [40, 50] \wedge Weight \in [65, 72] \wedge Gender \in \{F\} \wedge BG \in \{A, B, AB\}) \text{ then } Diagnosis = Sick$$

is inconsistent because the object trn_3 satisfies the premise of the rule and has a different decision.

Again, it should be noted that for g-rule we obtain an analogous relationship of the set $MaxRules(GenRules, trnSet)$ and *g-rule* (tst, trn) to the relation between $MaxRules(SimRules, trnSet)$ and *s-rule*(tst, trn). This is expressed in the following lemma.

Lemma 3.3. *Let tst be any test object and trn be any training object. Let GenRules be defined by parameters* ϱ_a *(from given pseudometric decision system) and* $c_a = a(tst)$ *for* $a \in A_{sym}$ *(see Definition 2.8), i.e.* $GenRules = GenRules\left(\{(\varrho_a, a(tst))\}_{a \in A_{sym}}\right)$. *Then any rule* $r \in MaxRules(GenRules, trnSet)$ *covering objects tst and trn is implied by the rule g-rule* $\left(tst, trn, \{\varrho_a\}_{a \in A_{sym}}\right)$.

Proof. The proof is an extension of proof of Lemma 3.1. For numerical attributes, the proof is the same as before.

For symbolic attributes, we substitute the part of the proof of Lemma 3.1 by the following one. Let us assume that a is symbolic. Then $t_a(r)$ is of the form $a \in B(a(tst), R_a)$, where $R_a = \varrho_a(a(tst), v)$, for some $v \in V_a$. Hence, because $t_a(r)(trn)$ is satisfied, then $R_a \geq \varrho_a(a(tst), a(trn))$. Thus $B\left(a(tst), r_a\right) \subseteq B(a(tst), R_a)$, where $r_a = \varrho_a(a(tst), a(trn))$. Then, $t_a(\text{g-rule}\,(tst, trn)) \Rightarrow t_a(r)$. □

Theorem 3.4. *Under the assumptions of Lemma 3.3, the rule g-rule* $\left(tst, trn, \{\varrho_a\}_{a \in A_{sym}}\right)$ *is consistent with the training set trnSet if and only if there exists a rule from* $MaxRules(GenRules, trnSet)$ *covering the objects tst and trn.*

Proof. The proof is a modification of the proof of Theorem 3.2.

We substitute the induction step of the proof of Theorem 3.2 for the case when a_i is symbolic by the following one. Let us consider the case when a_i is symbolic. Then $t_i(r_{i-1})$ is of the form $a \in B(a(tst), r_a)$, where $r_a = \varrho_a(a(tst), v)$, for some $v \in V_a$. We consider possible extensions of this condition, i.e. conditions of the form $a \in B(a(tst), R_a)$, where $R_a = \varrho_a(a(tst), w)$, for some $w \in V_a$ such that $R_a \geq r_a$ and the consistency of the rule is preserved (with $trnSet$). Because V_a is finite, then there is a finite number of possible selections, and we choose the one with the maximal value of R_a. The chosen extension is maximally general.

When a_i is numerical, we make the extension of the formula as in Theorem 3.2.

Analogously as in Theorem 3.2, one can conclude that the last rule r_m is consistent with $trnSet$ and maximally general.

Analogously as in Theorem 3.2, we obtain (using Lemma 3.3) that if the rule $g\text{-}rule\,(tst, trn)$ is inconsistent with the training set $trnSet$, then there is no rule in $MaxRules(GenRules, trnSet)$ covering tst and trn.

□

Let us note that the set $MaxRules(GenRules, trnSet)$ is defined for the given values c_a for $a \in A_{sym}$ (during the testing procedure, for the test example tst, we assume $c_a = a(tst)$). The idea behind the set $MaxRules(SimRules, trnSet)$ was to compute all maximally general rules in advance to use it later during the classification process. In order to compute an analogous set $MaxRules(GenRules, trnSet)$ in advance, this should be done for all possible combinations of all possible values for all symbolic attributes. It would increase the number of generated rules by the factor no more than b^k, where b is the maximal cardinality of $|V_a|$ for $a \in A_{sym}$ and k is the number of symbolic attributes.

It is not obvious how to define local decision rules and maximally general rules independently of such given values to keep the analogous relation between general rules and local decision rules and to enable grouping of symbolic values using given pseudometrics for symbolic attributes. For example, one can consider defining local decision rules as in Definition 3.2 and next, redefining $GenRules$ in such a way that admissible group of values are balls centred at a value of an attribute and with all possible distances between values, i.e. admissible conditions would be of the form $a \in B(v, \varrho_a(v, w))$, where $v, w \in V_a$. However, it can be shown that such a procedure will not lead to the desired relation between $MaxRules(GenRules, trnSet)$ and g-rules (see Appendix A).

Theorem 3.4 shows that instead of computing the support sets for rules from $MaxRules(GenRules, trnSet)$ covering a new test case, it is sufficient to generate the g-rules for all training examples and then check their consistency with the training set. This is realised by the lazy Rule Induction Algorithm (RIA) presented below.

The function $isConsistent(r, verifySet)$ was presented in Subsection 2.3.2 in Algorithm 1. The RIA algorithm (see Algorithm 4) differs from the LAZY algorithm (see Algorithm 2) only in line 7 where instead of the rule $s\text{-}rule(tst, trn)$, the rule $g\text{-}rule\left(tst, trn, \{\varrho_a\}_{a \in A_{sym}}\right)$ is used.

From Theorem 3.4, one can conclude that the RIA algorithm computes the measure $Strength$ for $MaxRules = MaxRules(GenRules, trnSet)$. Thus, the results of the mentioned algorithm are equivalent to the results of the algorithm

Algorithm 4: RIA(tst, $trnSet$, $\{\varrho_a\}_{a\in A_{sym}}$)

Input: test example tst, training set $trnSet$, family of pseudometrics for symbolic attributes $\{\varrho_a\}_{a\in A_{sym}}$
Output: predicted decision for tst

```
1  begin
2      foreach decision v ∈ V_d do
3          supportSet(v) = ∅
4      end
5      foreach trn ∈ trnSet do
6          v = d(trn)
7          if isConsistent(g-rule(tst, trn, {ϱ_a}_{a∈A_sym}), trnSet) then
8              supportSet(v) = supportSet(v) ∪ {trn}
9          end
10     end
11     return arg max_{v∈V_d} |supportSet(v)|
12 end
```

based on calculating $MaxRules$ and using the measure $Strength$ as a strategy for conflict resolution. Hence, we have the corollary analogous to Corollary 2.3 (see Subsection 2.3.2).

Corollary 3.5. *For any test object tst, and the classifier from Equation 2.12 with $MaxRules = MaxRules(GenRules, trnSet)$, where $GenRules = GenRules\left(\{(\varrho_a, a(tst))\}_{a\in A_{sym}}\right)$, we have*

$RIA(tst, trnSet, \{\varrho_a\}_{a\in A_{sym}}) = decision_{MaxRules}(tst)$.

The time complexity of the RIA algorithm is the same as that of the LAZY algorithm (see Algorithm 2), i.e. $O(mn^2)$, where $n = |trnSet|$, $m = |A|$. Again, this is far more efficient than generating the set $MaxRules(GenRules, trnSet)$ in advance, which can be exponentially large relative to the number of training examples.

However, for data sets with quite a large number of examples, this time complexity is still too high to be used in practice. Hence, one of the motivations behind our work was also to reduce this complexity. As a result, some modifications to this algorithm were developed. This issue is discussed in more detail in Section 3.3.

3.3 Combining instance-based learning and rule methods – RIONA

From now on, we additionally assume that aggregation function Agr is defined by Equation 2.1 unless it is stated differently.

The RIONA algorithm is based on a combination of instance-based learning and rule methods. The primary component of the RIONA algorithm was developed using the observation that the kNN method (see Subsection 2.3.3) is widely used, and usually for small values of k, it has quite good performance. On the other hand, in the

case of rule-based methods, in general, all training examples are used in the process of rule generation. Based on the observation related to the kNN method, one may suppose that only close training examples to the test case are important in the process of inducing the final decision. In fact, in the RIONA algorithm, the classification of the given test example is based on training objects from a neighbourhood of this example.

Thus, instead of considering all training examples in constructing the support set, like in the RIA algorithm, one can bound it to a certain neighbourhood of a test example. The intuition behind this is that the training examples which are far from a given test object are less relevant for classification than the closer ones.

We consider the neighbourhood N defined in Subsection 2.3.3. This neighbourhood is analogous to the one used in the kNN algorithm (i.e. the same as in the specific kNN algorithm defined in Subsection 2.3.3; see Algorithm 3).

Now, we are ready to present an approach to inducing of decision that is a kind of combination of instance-based learning (see Subsection 2.3.3) and lazy rule learning (see Section 3.2). The main idea is based on the strategy for conflict resolution with the use of *Strength* measure (see Equation 2.11) slightly modified in the following way:

$$LocalStrength(tst, v, k, \varrho) = \left| \bigcup_{r \in MatchRules(tst,v)} localSupportSet(r) \right|, \tag{3.1}$$

where most notation remains the same as in Equation 2.11; $\varrho = Agr(\{\varrho_a\}_{a \in A})$ is the aggregated pseudometric and k is the number indicating the size of the neighbourhood, $localSupportSet(r) = supportSet(r) \cap N(tst, trnSet, k, \varrho)$. The difference lies in the fact that we consider only those examples covered by the rules matched by a test object that are in a specified neighbourhood of the test example. The predicted decision based on the measure *LocalStrength* is analogous to the previous one (see Equation 2.12):

$$decisionLocal_{MaxRules}(tst, k, \varrho) = \arg\max_{v \in V_d} LocalStrength(tst, v, k, \varrho). \tag{3.2}$$

In the classification process, we assume that number k for the neighbourhood $N(tst, k)$ is fixed. The proper size of the neighbourhood, i.e. the parameter k is found in the learning phase (see Section 3.4).

Given a set $MaxRules$, the above measures can be calculated by limiting the support sets of the rules covering a test example to the specified neighbourhood of a test example. Thus the algorithm based on maximally general rules with the modified measure *LocalStrength* can be used here.

However, the measure *LocalStrength* can also be calculated using the lazy rule learning methodology. This is analogous to the fact that the RIA algorithm computes the measure *Strength* (see Corollary 3.5). To implement this approach, we modified Algorithm 4. First, in line 5 of the algorithm, only examples $trn \in N(tst, k)$ should be considered. Furthermore, it is not necessary to consider all the examples from the

training set to check the consistency of the *g-rule* $\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}\right)$ (see line 7 of Algorithm 4). This is due to the following proposition.

Proposition 3.6. *Suppose that* ϱ_a *in pseudometric decision system for* $a \in A_{num}$ *are defined as in Equation 2.2. If* $trn' \in trnSet$ *satisfies g-rule* $\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}\right)$, *then* $\varrho(tst, trn') \leq \varrho(tst, trn)$, *where* $\varrho = Agr(\{\varrho_a\}_{a\in A})$ *and the aggregation function Agr is defined either by Equation 2.1 or Equation 3.3.*

Proof. If trn' satisfies *g-rule* $\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}\right)$, then from the definition of g-rule (see Definition 3.2), we have:

- for all symbolic attributes $a \in A$: $a(trn') \in B\left(a(tst), r_a\right)$, where $r_a = \varrho_a(a(tst), a(trn))$. Then from definition of the closed ball we have $\varrho_a(a(tst), a(trn')) \leq \varrho_a(a(tst), a(trn))$.
- for all numerical attributes $a \in A$: $a(trn') \in [min_a, max_a]$, where $min_a = min(a(tst), a(trn))$, $max_a = max(a(tst), a(trn))$. Then, we have $|a(tst) - a(trn')| \leq |a(tst) - a(trn)|$. Thus, using definiton of metric for numerical attributes (see Equation 2.2) we have $\varrho_a(a(tst), a(trn')) = \frac{|a(tst)-a(trn')|}{a^{\max}-a^{\min}} \leq \frac{|a(tst)-a(trn)|}{a^{\max}-a^{\min}} = \varrho_a(a(tst), a(trn))$.

It means that for all attributes $a \in A$ we have $\varrho_a(a(tst), a(trn')) \leq \varrho_a(a(tst), a(trn))$. In consequence, we have the inequality between the global distances[2] for Agr defined by Equation 2.1: $\varrho(tst, trn') = \sum_{a\in A} \varrho_a(a(tst), a(trn')) \leq \sum_{a\in A} \varrho_a(a(tst), a(trn)) = \varrho(tst, trn)$.

Adding multiplication factor (specified by a weight) for each attribute preserves the above inequality. In consequence, we have the same result also for aggregation function defined by Equation 3.3.

□

Hence, the examples that are distanced from the test example tst more than the training example trn cannot cause inconsistency of *g-rule* $\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}\right)$. In consequence, we can substitute the set $trnSet$ in line 7 of Algorithm 4 by $N(tst, trnSet, k, \varrho)$. We can restrict even more the set of examples which can cause inconsistency (see Subsection 3.3.3).

The resulting classification algorithm RIONA is presented in Algorithm 5. The formal proof that this algorithm computes measure *LocalStrength* is presented later in Theorem 3.10. This algorithm predicts the most common class among the training examples that are covered by the rules satisfied by a given test example and are in the specified neighbourhood. It is to be noted that in the argument of the algorithm, all pseudometrics are given (used for computation of the final pseudometric), but in the g-rule only pseudometrics for symbolic attributes are used (see Definition 3.2 and note after it).

[2]It should be observed that we use in the proof the assumption that pseudometrics used for grouping symbolic attributes are the same as the pseudometrics which compose the aggregated pseudometric used for measuring distance between examples. The analogous assumption is used for numerical attributes: real values are grouped using interval contained in the ball $B(a(tst), \varrho_a(a(tst), a(trn)))$ determined by the Euclidean metric. The same Euclidean metric (however normed) is used for components of the final pseudometric.

Algorithm 5: RIONA-classify(tst, $trnSet$, k, $\{\varrho_a\}_{a\in A}$)

Input: test example tst, training set $trnSet$, positive integer k, family of pseudometrics for attributes $\{\varrho_a\}_{a\in A}$
Output: predicted decision for tst

```
1  begin
2      ϱ = Agr({ϱ_a}_{a∈A})
3      neighbourSet = N(tst, trnSet, k, ϱ)
4      foreach decision v ∈ V_d do
5          supportSet(v) = ∅
6      end
7      foreach trn ∈ neighbourSet do
8          v = d(trn)
9          if isConsistent(g-rule(tst, trn, {ϱ_a}_{a∈A_sym}), neighbourSet) then
10             supportSet(v) = supportSet(v) ∪ {trn}
11         end
12     end
13     return arg max_{v∈V_d} |supportSet(v)|
14 end
```

For every decision class, the RIONA algorithm computes the support set restricted to the neighbourhood $N(tst, k)$ rather than the whole support set of the maximally general rules covering a test object (as the RIA algorithm) in the following way. For every training object trn from the neighbourhood $N(tst, k)$ the algorithm constructs the rule $g\text{-}rule\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}\right)$ based on the considered example trn and the test example tst. Then, it checks whether this g-rule is consistent with the remaining training examples from the neighbourhood $N(tst, k)$. If the local decision rule is consistent, then the training example trn used to construct the rule is added to the support set of the appropriate decision. Finally, the algorithm selects the decision with the support set of the highest cardinality.

3.3.1 Some specific situations

Tie-breaking procedure of RIONA

During selecting decisions based on Equation 3.2, sometimes more than one decision class with the same biggest value of measure $LocalStrength(tst, v, k, \varrho)$ is obtained. In this case, in order to guarantee deterministic behaviour of our algorithm, it should be specified a tie-breaking procedure for selecting decisions. For clarity of the RIONA algorithm presentation, we did not place this procedure in Algorithm 5. However, we implemented it in such a way that the final decision is calculated dynamically for the consequent sizes of the neighbourhood up to the parameter k. In consequence, for our implementation, the tie-breaking procedure is applied in the following order: decision class with the biggest value of measure $LocalStrength(tst, v, k, \varrho)$, decision class with the biggest value of measure $LocalStrength(tst, v, k-1, \varrho)$, ..., decision class with the biggest value of measure $LocalStrength(tst, v, 1, \varrho)$, majority decision

class, i.e. decision class with the maximal value $|Class(v)|$, decision class with the smallest index.

Inconsistencies in data sets

We assumed that training sets are consistent (see Section 2.1). This simplifies the notation and proofs. However, RIONA works also with inconsistent training sets. For instance, if there are two examples with all conditional attributes with the same values and with different decisions, then any g-rule covering theses examples will be inconsistent. Inconsistencies in training set may cause that for a given test example, all support sets for all decisions are empty. We somehow improve this drawback in the case of the RIONIDA algorithm (see Subsection 4.3.6). The idea presented there could be also applied for the RIONA algorithm.

Missing values

We also assumed that there are no missing attribute values for any object (see Section 2.1). This also simplifies the notation and proofs. The problem of missing values is not the one on which this thesis is focused on. However, in RIONA, a heuristic for dealing with missing values in data sets is implemented. Hence, it was also possible to test RIONA on data sets with missing values. Now, we briefly explain how in RIONA pseudometrics are calculated and consistency of g-rule is checked when examples with missing values occur.

First, we need to extend the definition of pseudometrics. In Equation 2.1 occur pseudometrics ϱ_a for any attribute $a \in A$. Up to now, these pseudometrics were not defined for missing values. We also define $\varrho_a(v, w)$ for the case when v or w is a missing value assuming that in such a case ϱ_a takes the maximal possible value. Thus, it is equal to 1 for numerical attribute (which corresponds to the distance between the minimal and maximal value for the given attribute in the training set); and it is equal to 2 for any symbolic attribute (which corresponds to the theoretical maximal distance between a pair of symbolic values[3]). Hence, the distance between any value and missing value is not less than the distance between arbitrary two known values. In consequence, in the neighbourhood will be considered as more preferred, the objects having as many as possible known values. Such an approach could be interpreted as a conservative where we assume that it is better to reject the close object (to the tested object) with missing value(s) than to make a mistake by treating an object as close which actually is not such.

Second, if missing values occur, we need to define how the consistency of g-rule (see Definition 3.2) is checked. Let us fix attribute $a \in A$. If any elementary condition t_a (see Definition 2.6) contains missing value ($trn(a)$ or $tst(a)$ is missing), then all objects satisfy this condition, i.e. the semantics of t_a is equal to $\mathbf{X}$. If the elementary condition does not contain missing value, then objects with missing value for the considered attribute does not satisfy this condition.

The above-described approach to treating missing values is roughly based on the interpretation of missing value for a given attribute as any possible value from the set of values of this attribute.

[3]The actual maximal distance between symbolic values in a given training set can be smaller.

3.3.2 Time complexity of RIONA for the testing phase

Theorem 3.7. *Time complexity for the testing phase of RIONA is* $O(m(n+|N|^2))$ *for a single test object, where* $n = |trnSet|$, $m = |A|$, k *is the parameter used to define the size of neighbourhood,* $|N| = |N(tst,k)|$ *is the actual size of the neighbourhood for the given test object.*

Proof. In any run of the RIONA algorithm for a single test object, two phases can be distinguished.

In the first phase, training examples from the neighbourhood N are selected, i.e. k nearest objects to the test example (or more objects in the specific situation described in Definition 2.14) among n objects, where $n = |trnSet|$. It can be done in the linear time with respect to n (see e.g. [43]). Taking into account that for any object all attributes should be examined, time complexity of this phase is $O(mn)$, where $m = |A|$.

In the second phase, the algorithm checks consistency among objects from the neighbourhood N. Thus, it performs $O(m|N|^2)$ operations.

To sum up, the time complexity of the RIONA algorithm for the testing phase is $O(m(n+|N|^2))$ for a single test object.

□

Theoretically, $|N|$ can be equal to the size of the training set. However, for almost all tested data sets, $|N| = |N(tst,k)| \leq c \cdot k$ for all test examples tst, where c is a constant very close to 1[4]. Hence, for simplicity, we assume that $|N|$ is bounded in this way.

Corollary 3.8. *Assume that* $|N| = |N(tst,k)| \leq c \cdot k$ *for any test example* tst*, where* c *is a constant very close to 1. Then time complexity for the RIONA testing phase is* $O(m(n+k^2))$ *for a single test object, where* $n = |trnSet|$, $m = |A|$, k *is the parameter used to define the size of neighbourhood.*

In the case when k is treated as a constant[5] parameter of the algorithm (independent of n), the time complexity of RIONA (the testing phase for single test object) is $O(mn)$, while the time complexity of RIA is $O(mn^2)$ which gives us a significant acceleration for RIONA in comparison with RIA.

3.3.3 Further acceleration of RIONA

The set of examples to be searched for causing inconsistency can be restricted even more in comparison to what was discussed after Proposition 3.6 and presented in Algorithm 5. The details based on Proposition 3.6 are presented below.

First, let us assume that examples from $N(tst,k) = N(tst,trnSet,k,\varrho)$ (where the parameters have the meaning as discussed before) have different distances

[4] The exception is, for example, *mammography* data set consisting of many objects with the same value for any conditional attribute. However, in this case one could consider special data structures for grouping objects with identical attribute values for speeding up searching for the neighbourhood N.

[5] In the case when k is dependent on n it is sufficient to assume that $k < \sqrt{n}$ to keep the conclusion.

from the test example tst. Let $N(tst,k) = \{nn_1, nn_2, \ldots nn_k\}$, where nn_i is the i-th nearest neighbour of tst (i.e. training examples trn from $N(tst,k)$ are sorted by values $\varrho(tst,trn)$). From Proposition 3.6, we have that inconsistency of the rule $g\text{-}rule\,(tst,nn_i)$ can be caused only by the examples $nn_1, nn_2, \ldots, nn_{i-1}$ (i.e. closer examples to tst than the example nn_i). Hence, for the training example $nn_i \in N(tst,k)$, only $i-1$ training examples have to be checked whether they cause inconsistency. In this case, checking consistency requires $m(0+1+\cdots+(k-1))$ operations, where $m = |A|$. This gives the time complexity $O(mk^2)$.

Second, let us consider the general case when some examples from $N(tst,k)$ may have the same distances from the test example tst. In particular, then $N(tst,k)$ may contain more than k examples. For training example $nn_i \in N(tst,k)$ not only $nn_1, nn_2, \ldots, nn_{i-1}$ examples should be checked whether they cause inconsistency, but also such examples $nn_{i+1}, \ldots, nn_{i+l}$, for which $\varrho(tst,nn_i) = \varrho(tst,nn_{i+1}) = \ldots = \varrho(tst,nn_{i+l})$ holds for some number l.

In fact, all these cases were taken into account in our implementation. However, it does not change the order of overall time complexity. Due to this fact and for simplicity, we presented in Algorithm 5 a simplified version of the RIONA algorithm.

3.3.4 Relationships of RIONA to other approaches

Algorithm 5 (RIONA) is based on a combination of the kNN method with lazy rule induction. The only difference in comparison to Algorithm 3 (kNN) is in line 9, where we check the consistency of rule generated by training and testing example. One can say that the consistency of training example trn is checked. Thus we can interpret this as assigning the weight zero to inconsistent examples. This is compatible with the idea of instance-based learning paradigm (see Subsection 2.3.3) enriched by adding specification of weights for examples. In this sense, one can regard the RIONA algorithm as an instance-based algorithm.

We have the following relationships between RIONA, RIA and kNN.

Proposition 3.9. *For each test object tst*

$$RIONA(tst,trnSet,k,\{\varrho_a\}_{a\in A}) = \begin{cases} RIA(tst,trnSet,\{\varrho_a\}_{a\in A_{sym}}) & \text{for } k \geq |trnSet| \\ 1NN(tst,trnSet,\varrho) & \text{for } k = 1, |N(tst,trnSet,k,\varrho)| = 1, \end{cases}$$

where 1NN *is the nearest neighbour algorithm for* $k = 1$ *for pseudometric* $\varrho = Agr(\{\varrho_a\}_{a\in A})$.

Proof. For $k \geq |trnSet|$, the *neighbourSet* in the RIONA algorithm (see Algorithm 5) is equal to $trnSet$. In this case, the RIONA algorithm works exactly as the RIA algorithm (see Algorithm 4).

For $k = 1$, $|N(tst,trnSet,1,\varrho)| = 1$, the *neighbourSet* in the RIONA algorithm contains one training example[6]. In this case, consistency checking procedure could

[6]Please note that the assumption about the cardinality of N is important. When $|N| > 1$, even for consistent training set, examples from the neighbourhood N (equally distanced from test example) will cause inconsistency whenever there are two objects in N with different decisions.

be eliminated. In consequence, the RIONA algorithm works exactly as 1NN (see Algorithm 3). □

For the maximal neighbourhood, the RIONA algorithm works exactly as the RIA algorithm (and from Corollary 3.5 as the algorithm based on the maximally general rules with *Strength* as a strategy for conflict resolution). On the other hand, taking a neighbourhood consisting of the single nearest training example, we obtain the nearest neighbour algorithm. In this sense, RIONA is placed between the nearest neighbour classifier and the classifier based on maximally general rules. The choice of a small neighbourhood causes the algorithm to behave more like kNN classifier. The choice of a large neighbourhood causes the algorithm to behave more like a classifier based on inducing maximally general rules. Taking a larger but not the maximal neighbourhood can be seen as considering more specific rules instead of maximally general rules consistent with the training examples.

Now, we also show that we can look at the RIONA algorithm from other perspectives.

Theorem 3.10. *For any test object tst, the classification result of the classifier from Equation 3.2 with* $MaxRules = MaxRules(GenRules, trnSet)$*, where* $GenRules = GenRules\left(\{(\varrho_a, a(tst))\}_{a\in A_{sym}}\right)$*,* $\varrho = Agr(\{\varrho_a\}_{a\in A})$*, we have*

$RIONA(tst, trnSet, k, \{\varrho_a\}_{a\in A}) = decisionLocal_{MaxRules}(tst, k, \varrho)$.

Proof. From Corollary 3.5, $RIA(tst, trnSet, \{\varrho_a\}_{a\in A_{sym}}) = decision_{MaxRules}(tst)$. In fact, as we noticed it is implied from more strong fact following from Theorem 3.4 that the RIA algorithm computes measure *Strength* for $MaxRules = MaxRules(GenRules, trnSet)$, i.e. for each $v \in V_d$ $supportSet(v)$ (in line 11 of Algorithm 4) $= Strength(v)$. The RIONA algorithm takes into account only training examples from $N(tst, trnSet, k, \varrho)$. Moreover, from Proposition 3.6, examples consistent with $N(tst, trnSet, k, \varrho)$ are consistent with the whole training set, *trnSet*. At the same time, measure $LocalStrength(tst, v, k, \varrho)$ takes into account only training examples from the same neighbourhood $N(tst, trnSet, k, \varrho)$. In consequence, for each $v \in V_d$ $supportSet(v)$ (in line 13 of Algorithm 5) $= LocalStrength(tst, v, k, \varrho)$. This implies the equation of the theorem. □

Theorem 3.11. *For any test object tst, the classification result of the classifier from Equation 3.2 with* $MaxRules = MaxRules(GenRules, N(tst, trnSet, k, \varrho))$*, where* $GenRules = GenRules\left(\{(\varrho_a, a(tst))\}_{a\in A_{sym}}\right)$*,* $\varrho = Agr(\{\varrho_a\}_{a\in A})$*, we have*

$RIONA(tst, trnSet, k, \{\varrho_a\}_{a\in A}) = decisionLocal_{MaxRules}(tst, k, \varrho)$.

Proof. From Theorem 3.10 with *trnSet* replaced by $N(tst, trnSet, k, \varrho)$ (it is in a sense treated as a new training set), we have $RIONA(tst, N(tst, trnSet, k, \varrho), k, f) = decisionLocal_{MaxRules}(tst, k, \varrho)$, where $f = \{\varrho_a\}_{a\in A}$, $MaxRules = MaxRules(GenRules, N(tst, trnSet, k, \varrho))$. Since $RIONA(tst, N(tst, trnSet, k, \varrho), k, f) = RIONA(tst, trnSet, k, f)$, this ends the proof.

□

To sum up, these two theorems give the following interesting corollary.

Corollary 3.12. *For any test object tst, the results computed by the following classifiers are the same*

1. $RIONA(tst, trnSet, k, \{\varrho_a\}_{a\in A})$,
2. $decisionLocal_{MaxRules}(tst, k, \varrho)$,
3. $decisionLocal_{MaxLocalRules}(tst, k, \varrho)$,
4. $decision_{MaxLocalRules}(tst)$ *for new training set* $trnSet' = N(tst, trnSet, k, \varrho)$,

where $\varrho = Agr(\{\varrho_a\}_{a\in A})$, $MaxRules = MaxRules(GenRules, trnSet)$, $MaxLocalRules = MaxRules(GenRules, N(tst, trnSet, k, \varrho))$, *and* $GenRules = GenRules\left(\{(\varrho_a, a(tst))\}_{a\in A_{sym}}\right)$.

Proof. Equivalence of first three classifiers is implied directly by Theorems 3.10 and 3.11. It remains to prove equivalence of third and fourth classifiers. Note that $trnSet' = N(tst, trnSet, k, \varrho) = N(tst, N(tst, trnSet, k, \varrho), k, \varrho) = N(tst, trnSet', k, \varrho)$. Always holds $supportSet(r) \subseteq trnSet'$. So using previous equation $supportSet(r) \subseteq N(tst, trnSet', k, \varrho)$. Then $localSupportSet(r) = supportSet(r) \cap N(tst, trnSet', k, \varrho) = supportSet(r)$. We conclude that Equation 3.1 becomes Equation 2.11, and finally that Equation 3.2 becomes Equation 2.12.

□

In other words, we have the following conclusions. First, the RIONA algorithm computes the *LocalStrength* measure (second algorithm above; see Equation 3.1). Second, the *LocalStrength* measure is simply the *Strength* measure treating $N(tst, k)$ as the local training set (fourth algorithm). This fourth algorithm is the same algorithm as in Equation 2.12 with training set *trnSet* replaced by local training sample $trnSet' = N(tst, trnSet, k, \varrho)$. Therefore the RIONA algorithm can be treated as an algorithm for computing all maximally general, consistent rules locally and using (locally) *Strength* for conflict resolution. In Table 3.1, a general comparison of these three algorithms is presented (we omit the third algorithm, which is very similar to the fourth). In Table 3.2, a comparison scheme for these three algorithms is presented (with explanation what is common and what is different in these algorithms).

RIONA	algorithm (2) based on the measure *LocalStrength*	algorithm (4) based on the measure *Strength* counted locally
	counting rules	
no need to count rules explicitly	counts $MaxRules$ globally once at the beginning	counts $MaxRules$ locally for each test case
	counting support	
counts support using lazy local rules	counts support locally	counts support locally

Table 3.1: A general comparison of three algorithms from Corollary 3.12: algorithm (1) RIONA, algorithm (2) based on the measure *LocalStrength* and algorithm (4) based on the measure *Strength* counted locally.

RIONA	algorithm (2) based on the measure *LocalStrength*	algorithm (4) based on the measure *Strength* counted locally
Global input: $trnSet$, $k \in \mathbb{N}$		
1.	count $MaxRules$ for $trnSet$	
Input: test case tst		
2. $nSet = N(tst, k)$		
3.	$RuleBase = MaxRules$	count (locally) $MaxRules(nSet)$ $RuleBase = MaxRules(nSet)$
4.	consider rules from $RuleBase$ with premise satisfied by tst	
5. for each decision d		
6. consider $trn \in nSet$ with decision d	consider rules from step 4 with decision d	
7. count the number of trn from step 6 forming consistent rules with tst	count the number of $trn \in nSet$ supporting rules from step 6	
8. choose the decision with the maximal count (maximally supported)		

Table 3.2: A comparison scheme of three algorithms from Corollary 3.12: algorithm (1) RIONA, algorithm (2) based on the measure *LocalStrength* and algorithm (4) based on the measure *Strength* counted locally.

3.3.5 RIONA and rules

The RIONA algorithm has properties of instance-based classifiers and rule-based classifiers. There are some aspects of rule-based classifiers which are more preferred for users than instance-based classifiers even at the expense of decreasing the quality of classification. One of these important aspects is the possibility to interpret rules by a human, non-computer science expert. He or she can verify whether the discovered knowledge in such rules is non-trivial, true in fact and revealing new aspects of the regarded problem. A rule contains an explanation for taking the particular decision easily understandable by a human.

We assume here that the parameter k in the RIONA algorithm is fixed (possibly learned as described in Section 3.4). Now, let us focus on algorithm (4) from the previous subsection. At first sight, the direct computation of $MaxLocalRules$ may seem very expensive and infeasible because for each test case tst it is necessary to compute the local complete set of consistent and maximally general decision rules. However, let us note that the size of the local training sample compared to the size of the whole training sample is significantly reduced from $n = |trnSet|$ to k (if we assume that the size of N is k; see previous notes related to this issue). Thus the total cost of computation of (global or local) $MaxRules$ is reduced from $O(2^n)$ to $O(m \cdot 2^k)$, where m is the number of test objects. We present this approach not

only from a theoretical point of view. Such an algorithm could be used when an explanation of the decision undertaken by the classifier is required. In this sense, the RIONA algorithm has features of quick lazy learning algorithm and rule algorithm, i.e. its parameters can be translated into rules.

Moreover, it seems possible to extend algorithm (4) to build all rules globally once at the beginning analogously to algorithm (2) from Corollary 3.12 with such difference that the rules would base on the local neighbourhood. Such rules would imitate the behaviour of the RIONA algorithm. There are some advantages of such an approach. Firstly, the explanation for the specific test object in the form of a set of rules could be given quickly. Secondly, all possible rules generated at the beginning could be verified whether the discovered knowledge is really useful.

We give here only an informal description of how such rules could be generated. The idea is similar to algorithm (4) from Corollary 3.12. We could simply treat each training example as a test example and generate $MaxRules$ locally for each training case. It could be seen as specific local reducts calculation (i.e. reducts calculated in the process of generation of maximally general rules for a given object; see e.g. [15, 17, 164, 225]). Normally, in constructing local reducts, discernibility should be preserved for objects with different decisions. Here, we would require that only objects with a different decision and distanced not more than k should be discerned.

3.4 Estimating the optimal neighbourhood size

Analogously to the case of kNN classifier, one can expect that different values of the parameter k can be relevant for different data sets. In fact, during the experiments (see [81]), we found that performance of the algorithm can significantly depend on the size of the chosen neighbourhood, and different size is appropriate for different data sets (for the detailed results the readers are referred to [81]). Therefore, in terms of Accuracy of the algorithm, it is important to find the optimal neighbourhood, i.e. the parameter k. Analogously as in case of kNN classifier, one can estimate the optimal value of this parameter. Here, one can consider two important questions: (1) How to learn the optimal value of the parameter k efficiently? (2) Can we use a bound on the maximal possible value of k in the process of searching for its optimal value? We present answers to these questions in the following two subsections.

3.4.1 Efficient learning of the optimal parameter k

Below we describe the algorithm for estimation of the optimal value k for the neighbourhood $N(tst, k)$. This can be done in an analogous way to searching for the optimal value k for the kNN method. The leave-one-out method is used on a training set to estimate the Accuracy of the classifier for different values of k ($1 \leq k \leq k_{max}$); then the value of k with the highest estimated Accuracy is selected. Applying it directly would require repeating leave-one-out estimation k_{max} times. However, using dynamic programming technique, we emulate this process in time comparable to the single leave-one-out test for k equal to the maximal possible value $k = k_{max}$. Below we present the algorithm implementing this idea.

Algorithm 6: getClassificationVector(trn, $trnSet$, k_{max}, $\{\varrho_a\}_{a\in A}$)

Input: currently considered example $trn \in trnSet$, training set $trnSet$, number k_{max}, family of pseudometrics for attributes $\{\varrho_a\}_{a\in A}$

Output: vector A of leave-one-out classification for trn for different values of parameter $k = 1, 2, \ldots, k_{max}$

```
1  begin
2      ϱ = Agr({ϱ_a}_{a∈A})
3      N = N(trn, trnSet \ {trn}, k_max, ϱ)
4      vector nn_1, ..., nn_|N| = N sorted according to the distance ϱ(trn, ·)
5      foreach decision v ∈ V_d do
6          decStrength[v] = 0
7      end
8      currentDec = the most frequent decision in trnSet
9      for k = 1 to |N| do
10         if isConsistent(g-rule (trn, nn_k, {ϱ_a}_{a∈A_sym}), N) then
11             v = d(nn_k)
12             decStrength[v] = decStrength[v] + 1
13             if decStrength[v] > decStrength[currentDec] then
14                 currentDec = v
15             end
16         end
17         A[k] = currentDec
18     end
19     return A
20 end
```

For a training example trn, the function $getClassificationVector(\ldots)$ (see Algorithm 6) finds k_{max} examples from $trnSet \setminus \{trn\}$ nearest to the example trn and sorts them according to the distance $\varrho(trn, \cdot)$ (i.e. we consider the pseudometric ϱ with the first argument fixed). It should be pointed out that as in the testing phase, it is necessary to consider the neighbourhood $N(tst, k_{max})$, which, in general, may contain more than k_{max} objects. Next, it returns the vector of decisions that the RIONA classifier would return for successive values of k. Algorithm 7 calls this routine for every training object, and then it selects the value k for which the global estimation of Accuracy is maximal.

Time complexity of the learning phase

As it was mentioned previously, the neighbourhood $N(tst, k_{max})$ may contain in general more than k_{max} objects. However, we assume (which is true in most of our experiments; see Subsection 3.3.2) that the size of the neighbourhood $N(tst, k_{max})$ during the learning phase is close to k_{max} analogously as in the testing phase, i.e. $|N| = |N(trn, k_{max})| \leq c \cdot k_{max}$ for all[7] $trn \in trnSet$, where c is a constant very close

[7] It would be even enough to assume that this bound is assured on average among all training examples.

Algorithm 7: findOptimalK($trnSet$, k_{max}, $\{\varrho_a\}_{a\in A}$)

Input: training set $trnSet$, number k_{max}, family of pseudometrics for attributes $\{\varrho_a\}_{a\in A}$
Output: optimal k

```
1 begin
2     foreach trn ∈ trnSet do
3         A_trn = getClassificationVector(trn, trnSet, k_max, {ϱ_a}_{a∈A})
4     end
5     for k = 1 to k_max do
6         estimatedAccuracy[k] = |{trn ∈ trnSet : d(trn) = A_trn[k]}|
7     end
8     return arg max_{1≤k≤k_max} estimatedAccuracy[k]
9 end
```

to 1[8].

Theorem 3.13. *Assume that* $|N| = |N(trn, k_{max})| \leq c \cdot k_{max}$ *for all* $trn \in trnSet$, *where* c *is a constant very close to 1. Then time complexity of the learning phase of RIONA is* $O(mn(n + k_{max}^2))$, *where* $n = |trnSet|$, $m = |A|$, k_{max} *is the parameter used to define the maximal size of neighbourhood to be analysed.*

Proof. In the run of the learning algorithm, one can distinguish, for each training object (see lines 2-4 of Algorithm 7), three phases (realised by Algorithm 6).

In the first phase, training examples from the neighbourhood N are selected, i.e. k_{max} nearest objects to the considered training example (or more objects in the specific situation described in Definition 2.14) among n objects, where $n = |trnSet|$. It can be done in the linear time relative to n (see e.g. [43]). Taking into account that for any object all attributes should be examined, time complexity of this phase is $O(mn)$, where $m = |A|$.

In the second phase, the algorithm sorts all selected objects from the neighbourhood N. Computing distances for objects from N takes $O(m|N|)$ steps (once for every object from N). Sorting (using computed distances) can be done in $O(|N| \log |N|)$ steps. Thus, this phase takes $O(m|N| + |N| \log |N|)$ steps.

In the third phase, the algorithm checks consistency among the selected objects. It takes $O(m \cdot |N|^2)$ steps.

From the assumption on the bound of the neighbourhood N, the second and third phases altogether take $O(m \cdot k_{max}^2)$ steps.

Thus, time complexity of *foreach* loop within lines 2-4 of Algorithm 7 is $O(n(mn + mk_{max}^2)) = O(mn(n + k_{max}^2))$.

Finally, for the whole training set, the algorithm computes leave-one-out Accuracy for each $1 \leq k \leq k_{max}$ (see lines 5-7 of Algorithm 7). It takes $O(nk_{max})$ steps.

Summing up, the time complexity of the learning algorithm is $O(mn(n + k_{max}^2) + nk_{max}) = O(mn(n + k_{max}^2))$. □

[8] This condition could be easily satisfied in general if the algorithm were rebuilt to choose deterministically only k examples in the neighbourhood.

From time complexity point of view, it would be efficient to choose such value of k_{max} that the component $O(mn^2)$ would be predominant in the complexity function above. This issue is discussed in more detail in Subsection 3.4.2.

Optimal Nearest Neighbour algorithm (ONN)

Note that by ignoring the consistency checking in the function $getClassificationVector(\ldots)$, we obtain the kNN algorithm with the selection of the optimal k. We call this classification algorithm Optimal Nearest Neighbour algorithm (ONN), and we used it in experiments for comparison with RIONA and other algorithms (see [81] for details). ONN classifies a new test object tst with the most frequent decision in the set $N(tst, k)$, where the number k is selected as in the algorithm described above.

3.4.2 Bound of the parameter k

Can we bound the maximal possible values of k in the process of searching for its optimal value? It was shown above that the time complexity of the learning algorithm is $O(mn(n + k_{max}^2))$. Thus, it would be efficient from the point of view of time complexity if $k_{max} < \sqrt{n}$, where $n = |trnSet|$. In this case, component $O(mn^2)$ in the learning phase would be predominant. It is sufficient to keep this inequality for large data sets. If we assume that large data sets are those with the size of the training set at least 40 000, then it is enough to consider $k_{max} = 200$. Next, an important question is how such setting could affect the quality of the RIONA classifier.

In order to answer this question, we performed the following experiments. Here, we only briefly describe the most important results of these experiments, more precisely described in [81]. We use in the description names of data sets coming from the UCI repository[9]. For the smaller sets (less than 4000 objects), experiments were performed for all possible values of k, and for the greater sets, the maximal value k was set to $k_{max} = 500$ (for the *nursery* data set we made the exception setting $k_{max} = 1000$). The classification Accuracy was measured for the leave-one-out method applied to the whole set. Figures 3.2, 3.3, 3.4, 3.5 present the dependency of classification Accuracy on the value of k for exemplary data sets.

For most of the tested data sets, we observed that while increasing k beyond a certain small value, the classification Accuracy was decreasing (see Figures 3.2, 3.3, 3.4). Experiments have shown that for most of the data sets, the results for the total or a relatively large neighbourhood are significantly worse than the results for the neighbourhood found by the RIONA algorithm. For the remaining data sets (*breast, census-income, nursery, primary, solar-flare*), the Accuracy becomes stable beyond a certain value k (see Figure 3.5).

For the former group of data sets, we examined the neighbourhood size (value of k) for which the maximum Accuracy was obtained. In the latter case, we examined both the value of k beyond which Accuracy remains stable and the fluctuations in Accuracy while increasing k. It was found during experiments that in most cases, the

[9] We do not give technical details for these data sets. Only *german* data set, used in further experiments, is described in Subsection 5.1.3 (it is identified as *credit-g*)

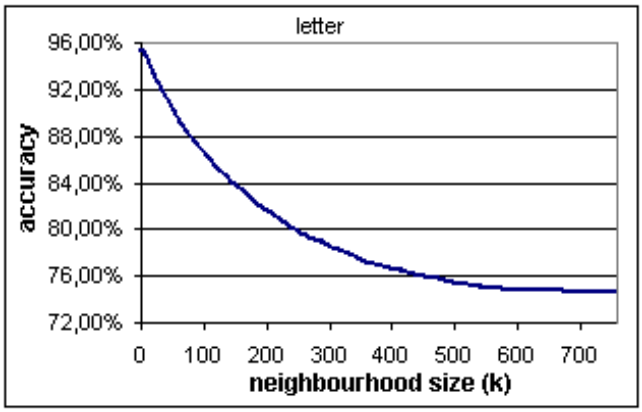


Figure 3.2: Classification Accuracy for *letter* data set.

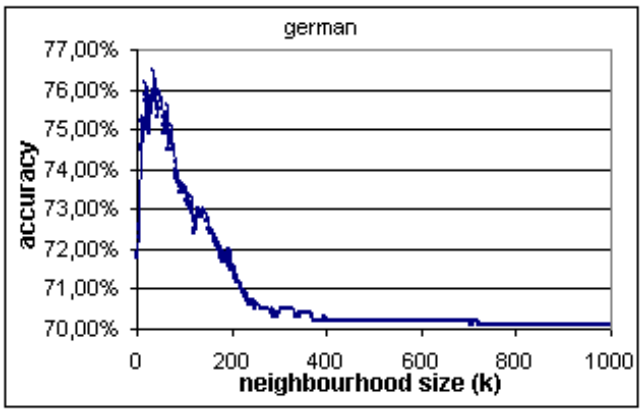


Figure 3.3: Classification Accuracy for *german* data set.

optimal value of k is less than 200. Moreover, for many data sets, the optimal value of k is less than 60, and for 7 of them, this value is equal to or less than 4. On the other hand, for the data sets where the optimal k was higher than 200 (*australian, census-income* and *nursery*), the loss in classification Accuracy related to this setting was insignificant: it remained within the range of 0.15%. Moreover, the Accuracy became stable for values of k much lower than 200. Therefore we could conclude that the setting $k_{max} = 200$ preserves good time complexity properties and does not significantly change the results for tested data sets.

The fact that a small neighbourhood gives the best Accuracy leads to another conclusion. Limiting the support set of a maximally general decision rule from $MaxRules$ to a neighbourhood of a test example can be seen as replacing the rule with a more specific one. In this sense, the presented results suggest that taking the complete set of consistent and maximally general decision rules usually gives worse Accuracy than a set of more specific rules. This is related to measures for conflict resolution taking into account the specificity of a rule as one of the important factors (see e.g. [87]).

For several data sets (*letter, pendigits, satimage, segment, shuttle* and *yeast*) we noticed that the Accuracy is falling down monotonically. Since for these data sets,

Figure 3.4: Classification Accuracy for *splice* data set.

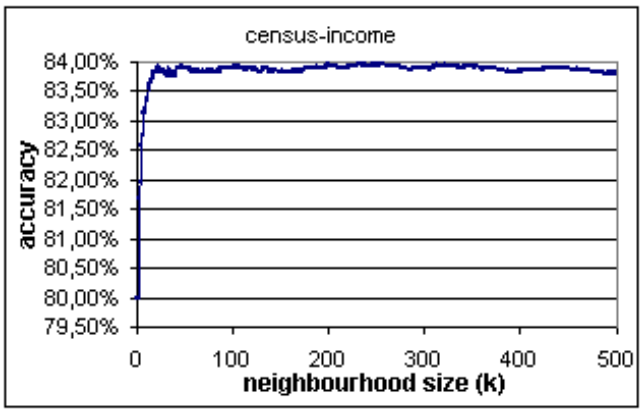


Figure 3.5: Classification Accuracy for *census-income* data set.

the best Accuracy is obtained for the smallest values of k, the kNN method seems to work best for them. On the other hand, all the mentioned data have numerical attributes. Hence, we can conclude that for numerical data, decisions are induced best by the kNN method and a falling down Accuracy characterises well the data sets that are appropriate for the kNN method.

If data are split into training and testing set one can ask the question whether the Accuracy on a test set obtained for the value k computed from a training set may differ significantly from the optimal Accuracy on a test set. In order to study this aspect, we compared these accuracies on the data sets that were originally split. The experiments showed that for *pendigits* Accuracy obtained by RIONA differs by about 0.5% from the Accuracy with the optimal number k and for other data sets the difference remains in the range of 0.2%. It means that the used algorithm finds almost the optimal number k in terms of the Accuracy obtained.

To sum up, there is no need to take the whole training set in the process of classification. Moreover, taking fewer objects can improve classification performance.

3.4.3 Comments on the structure of RIONA

Here, we present the general structure of RIONA. It consists of the training part and the classification part. Generally, these parts were presented previously. However, for clarity, we present these parts directly in the RIONA algorithm (see Algorithm 8). This brings all the details given previously together.

The main algorithm simply assigns the *options*. Furthermore, the aggregation of pseudometrics *Agr* (appearing in Algorithms 5, 6; see Subsection 2.2.3) by default is set up according to Equation 2.1. It may differ in the case when one selects the option with weights for attributes (see Subsection 3.6.3).

In the training part (function *RIONA-train*), the training set together with conditional and decision attributes are specified. Next, the pseudometrics for each attribute based on the training set are set up. Then, depending on given options other operations are done. Here, by default, the indexing tree for fast searching of the nearest neighbours is built (see Subsection 3.6.1). Finally, the procedure of searching for the optimal value of k is called and the result is stored in the local variable k_{opt} to be used later during classification.

Classification part (function *RIONA-classify*) simply calls the procedure RIONA-classify(...) (see Algorithm 5) using as parameters the given test example and other variables which were set up during the learning process.

Let us sum up the most important parts of the RIONA algorithm shown in Algorithm 8. During initialisation, RIONA defines *Agr*, i.e. the aggregations of pseudometrics (as default and usually used, the sum of pseudometrics for attributes). During training, pseudometrics for attributes are calculated and the optimal value of the parameter k is searched. These pseudometrics and the optimal value of the parameter k are used during classification.

It should be stressed that both computation of pseudometrics and searching for the optimal value k is always done using only the available training set (e.g. during the cross-validation process). This becomes clear from the description of Algorithm 8.

3.5 Experimental properties of RIONA

Numerous experiments were performed for the RIONA algorithm (see [81], see also [219]). Analogously to Subsection 3.4.2, we only briefly describe the most important results of the performed experiments, more precisely described in [81]. We use in the description names of data sets coming from the UCI repository.

The optimal size of a neighbourhood was estimated during the process of learning on the basis of the training examples. Before applying the algorithm no preprocessing was done. In particular, the discretisation of numerical attributes was not applied.

In this section, we describe some of the performed experiments and conclusions that led us to the final version of the presented algorithm. We also describe conclusions from experiments that can help us to understand the important features of RIONA.

Algorithm 8: RIONA(*options*)

Input: *options* (including k_{max}) of the RIONA algorithm (we do not list all of them here; see Section 3.6 for more details)

Global variables:
A – conditional attributes ($A = A_{num} \cup A_{sym}$)
d – decision attribute
Agr – the aggregation of pseudometrics (appearing in Algorithms 5, 6; see Subsection 2.2.3)
Local variables:
$trnSet$ – training set
$\{\varrho_a\}_{a \in A}$ – family of pseudometrics for attributes
k_{max} – the size of the neighbourhood to be used during searching for k_{opt}
... (local variables related to other options)
k_{opt} – optimal value for k
begin
    $k_{max} = options.k_{max}$
    by default Agr is defined according to Equation 2.1 (it may differ in case of choosing option for different weights for attributes – see Subsection 3.6.3)
    ... (assignments related to other options)
end
Function *RIONA-train(trnSetDescription) : void*
    Input: *trnSetDescription* – description of training set togeather with the specifiaction of decision and conditional attributes
    using *trnSetDescription* specify the conditional attributes A, the decision attribute d and the training set $trnSet$
    foreach $a \in A_{num}$ **do**
        ϱ_a =normalised city-block metric based on $trnSet$ (see Equation 2.2)
    end
    foreach $a \in A_{sym}$ **do**
        ϱ_a = SVDM pseudometric based on $trnSet$ (see Equation 2.4)
    end
    ...(operations related to other options)
    k_{opt} = findOptimalK($trnSet$, k_{max},$\{\varrho_a\}_{a \in A}$) (see Algorithm 7)
end
Function *RIONA-classify(tst) : decision*
    Output: predicted decision for tst
    return RIONA-classify(tst, $trnSet$, k_{opt}, $\{\varrho_a\}_{a \in A}$) (see Algorithm 5)
end

3.5.1 RIONA versus other algorithms and different settings for RIONA

We also compared RIONA with other learning algorithms and checked different settings for the RIONA algorithm. These experiments led us to the following conclusions.

Although during preliminary experiments, we have also tried other types of pseudometrics, no one appeared to be significantly better than the presented one in terms of Accuracy on a range of data sets. (In [219], a report from extensive experiments for different pseudometrics is presented.) We have also compared two measures as a strategy for conflict resolution: *LocalStrength* and *LocalStrength* normalised by the decision class size. We obtained almost identical results for both of them.

Apart from the neighbourhood N (see Definition 2.14), we considered also the neighbourhood B. It is defined as the set of objects with the distance from the test object bounded by a specified value. For this kind of neighbourhood, we applied an analogous idea to the neighbourhood N for estimating the optimal neighbourhood size. Instead of considering k_{max} successive values in the *for* loop (line 9) of Algorithm 6, relevant intervals for the radius R were considered. We studied both classes, i.e. N and B of a neighbourhood for the RIONA algorithm in parallel and the empirical difference between them was discussed in [81]. The experiments presented in [81] show that the neighbourhood N, in general, has better performance in terms of Accuracy. That is why in the thesis (this chapter and the next chapter for the RIONIDA algorithm) we focus only on this kind of neighbourhood.

The Accuracy of RIONA and ONN is comparable or better than that of the well-known learning algorithms: 3NN, C5.0, DeEPs and DeEPsNN. In particular, their Accuracy is generally better than the Accuracy of RIA and 3NN. This suggests the conclusion that RIONA and ONN may replace successfully both the rule-based algorithm using all maximally general rules and the kNN with a fixed k. The performed tests on numerous data sets indicate that the presented algorithm works well for data sets with both numerical and symbolic attributes. In particular, it works well for numerical attributes without preprocessing.

Some independent researchers used the publicly available version of RIONA (see Section 1.7) to compare many algorithms (including RIONA) for different real-life classification problems. As an example, RIONA was reported to obtain very good or good results in the following publications[10]: [48, Chapter 1] (RIONA was first on 21 tested algorithms), [84] (first on nine algorithms), [178] (second on eight algorithms), [85] (second on eight algorithms), [8] (fifth on 47 algorithms). Moreover, the authors of RIONA do not know any publication reporting that RIONA obtained low classification quality compared to some other algorithms. The mentioned results can be regarded as an argument (independent of the authors of RIONA) for applying

[10]However, it should be noted that the publicly available version of RIONA has different default settings than those used as default ones in this thesis. The different settings used as default in the publicly available version of RIONA (and used in the cited comparisons) are the following: inverse square distance as a voting method (see Subsection 3.6.2), distance-based as a weighting method (see Subsection 3.6.3), switch indicating whether nearest neighbours are filtered by rules is turned off (see Subsubsection 'Optimal Nearest Neighbour algorithm (ONN)' on page 84).

RIONA (or ONN) for real-life classification problems.

3.5.2 RIONA versus ONN

Experiments presented in [81] related to comparison of RIONA and ONN (kNN with selection of the optimal neighbourhood) showed that the significant differences in Accuracy occurred mostly for smaller data sets (*breast, bupa-liver, chess, primary, solar-flare* and *yeast*). Differences for all other data sets are less than 1%.

The only difference between RIONA and ONN is the operation of consistency checking. In other words, RIONA uses rules to filter nearest neighbours while ONN uses all nearest neighbours. In order to explain the similarity of results, we checked if using consistency checking changes substantially the set of examples taken into account (see [81]). The results presented in [81] showed that only for three data sets: *breast-cancer, primary* and *solar-flare* the operation of consistency checking eliminates a significant fraction of nearest neighbours. For other data sets the number of consistent objects from the optimal neighbourhood in the RIONA algorithm is close to the number of all objects from the optimal neighbourhood of the kNN algorithm. Therefore the differences in Accuracy are small. These observations suggest that the operation of consistency checking in the RIONA algorithm is not very significant (see Section 3.3).

We suspect these observations relate to the fact that usually the set of all consistent, maximally general rules is of a large size. The experiment carried out in [81] indicates that the support set induced from the whole set of consistent and maximally general rules contains a large fraction of all examples. On the other hand, the analysis of Accuracy in dependence on the number of neighbours k shows that usually a small number of objects gives the best Accuracy. It suggests that many of consistent and maximally general rules are induced rather randomly. Hence, considering either a reasonably computed smaller set of rules or a more restrictive operation of consistency checking may give better classification Accuracy.

3.6 Extensions of RIONA

Here, we briefly describe in what directions RIONA was extended in [219]. This shows some possible extensions of the RIONA algorithm, described in the thesis. But, more importantly, we used some of them to extend RIONIDA also. These variants can be used by setting relevant options of RIONA (see the initialisation part of RIONA described in Algorithm 8 from Subsection 3.4.3)

3.6.1 Indexing tree for fast searching for the nearest neighbours

Standard kNN methods store all the training examples to use it for classification of new unseen examples. One of the drawbacks of the standard kNN method is that for classifying new test example it is required to compute distances from all the stored training examples. This can cause slow testing speed for large data sets. To tackle this problem a special data structure can be used for fast searching of the

nearest neighbours. This was used in the algorithm presented in the thesis and used in experiments. Analogously, this data structure can speed-up RIONA during the learning phase. For more details of this solution, the readers are referred to [219].

3.6.2 Different types of voting

In Equation 3.1 all training objects from a $localSupportSet(r)$ of some rule r are counted with equal weight. Analogously to models proposed for kNN classifiers (see e.g. [7, 112, 149]) the RIONA classifier was adjusted to weight a vote of training example according to their distance to the test example. There are two implemented parameters of using weights for training example trn: $w_{trn} = \frac{1}{\varrho(tst,trn)}$ or $w_{trn} = \frac{1}{\varrho(tst,trn)^2}$. We used these developed methods (described in [219]) in our experiments for the RIONIDA algorithm as an extension of RIONA (see Subsection 5.5.5).

3.6.3 Different weights for attributes

In Equation 2.1 all attributes are treated as equally important. However, there are factors in real-life data sets which cause that different attributes have unequal significance. In fact, taking this into account can improve classification results (see e.g. [112, 151, 212]). Thus, Equation 2.1 can be replaced with its weighted version:

$$\varrho(x,y) = \sum_{a \in A} w_a \cdot \varrho_a(a(x), a(y)), \tag{3.3}$$

where $w_a \in \mathbb{R}$ is a weight for any attribute $a \in A$.

In [219], one can find methods for learning relevant weights as well as explanations for how the weighting algorithms could be applied for the RIONA algorithm. We used these developed methods (described in [219]) in our experiments for the RIONIDA algorithm as an extension of RIONA (see Subsection 5.5.5).

3.6.4 Extensions of SVDM pseudometric for numerical attributes

For symbolic attributes, we use in RIONA pseudometrics SVDM which are induced from the training set based on a correlation between attribute values and decision values. Such a correlation is not used in RIONA (by default) for numerical attributes. However, there were proposed pseudometrics for numerical attributes analogous to the SVDM pseudometric which help to overcome this drawback. In [219] two of them, namely *Interpolated Value Difference Metric* and *Density Based Value Difference Metric* are presented to extend RIONA. The readers are referred, e.g. to [219] for details and literature for these issues. We used these developed methods (described in [219]) in our experiments for the RIONIDA algorithm as an extension of RIONA (see Subsection 5.5.5). However, contrary to the normalised city-block metric (see Equation 2.2) proposition for these pseudometrics analogous to Proposition 3.6 may not hold[11]. In consequence, using these particular pseudometrics can cause that

[11] It is worth pointing out that RIONA could be rebuilt as mentioned in footnote 1 (page 68) to omit this problem.

inconsistent local rules can be recognised as consistent. Generally, if for numerical attributes other pseudometrics than metrics like Euclidean are used, then we, in fact, obtain somehow different algorithm. In particular, the resulting algorithm may not satisfy many of the presented above properties.

3.6.5 K nearest neighbours with local pseudometric induction

In the RIONA algorithm, we use some pseudometrics. These pseudometrics are induced globally during the learning phase, i.e. for the whole training set. In [219], a modification of this idea was introduced so that pseudometrics are induced locally, i.e. on the base of the neighbourhood of the test case. Thus, we have an extension of RIONA with local pseudometric induction. For details see [219]. This idea cannot be used directly for the RIONIDA algorithm. Thus, we did not use this extension in experiments for RIONIDA. However, this idea may be realised provided that its special adaptation for RIONIDA is developed.

3.7 Other possible extensions of RIONA

The RIONA algorithm, developed for balanced data, tries to optimise the common performance measure, i.e. Accuracy. However, RIONA could be extended in such a way that other performance measures than Accuracy could be used during optimisation. This issue is very important especially for imbalanced data and is discussed in the next chapter. In fact, RIONIDA implements this suggestion for performance measures dedicated to imbalanced data.

We presented the RIONA algorithm and its counterpart ONN. The latter algorithm is based on kNN. Experiments showed that the choice between these algorithms depends on the used data sets. For example, we found that generally, ONN performs better for data sets with only numerical attributes. Thus, these learning algorithms could be joined into one meta-learning algorithm which during the learning phase would select which one of them to choose and its optimal parameters. In fact, something like this, and even more was done for the RIONIDA algorithm (see next chapter).

3.8 Conclusions for RIONA

The research reported in this chapter attempts to bring together some ideas of instance-based learning and rule induction into a single algorithm.

First, we extended the LAZY algorithm presented in Chapter 2 for numerical attributes. In particular, the extended algorithm does not require discretisation. It groups values of numerical attributes into interval during lazy rule generation. Second, we also generalised the algorithm for grouping symbolic attributes. We showed the theoretical equivalence of this algorithm (RIA) with the algorithm generating all consistent and maximally general rules (in a specific set of rules).

Also, we (empirically) showed that for the correct classification of a test case, it is enough to consider only its small neighbourhood instead of the whole training set.

It illustrates the known empirical fact that while using rule-based classifiers, one can obtain better results by rejecting some rules instead of using all maximally general rules as the RIA algorithm does.

We found that the appropriate choice of the neighbourhood size is a crucial factor for obtaining high Accuracy. In this way, we proposed the RIONA algorithm using rules that are built based on a neighbourhood of the test case.

The fact mentioned above is also the key idea that allowed us to make the RIONA algorithm efficient without loss in Accuracy (compared to RIA). We also designed a method for efficient learning of the optimal size of the neighbourhood of RIONA. In practice, the complexity of learning and classification is only squarely and linearly dependent on the size of a learning sample, respectively. Although a great effort was put to speed up the presented algorithm, further acceleration was done, e.g. by more specialised indexing data structures (see [219]).

As the empirical results indicate, the presented algorithm obtained the Accuracy comparable to the well-known systems: 3NN, C5.0, DeEPs and DeEPsNN. The experiments showed that besides applying the newly proposed methods, a pseudometric choice is significant for the algorithm's Accuracy. Using pseudometric CSVDM proved to be very successful.

The facts that RIONA and ONN algorithms have similar Accuracy and the fraction of objects eliminated by the consistency checking operation is very small indicate that this operation has rather a small influence on the Accuracy of the presented algorithm. This suggests that the kNN component remains a dominant element of the presented algorithm and shows that either the construction of local decision rules should be more general or the operation of consistency checking should be more restrictive. Regardless of this fact, we preserved the algorithm's rule component (and in the next chapter, we prove its usefulness for imbalanced data).

Theoretical results show the RIONA classifier's relationships to both instance- and rule-based classifiers (see Subsections 3.2.2, 3.3.4 and 3.3.5). In particular, we showed the theoretical equivalence of the RIONA algorithm with the algorithm generating all consistent and maximally general rules in a training set consisting of the neighbourhood of the test case. Consequently, the RIONA classifier can be represented by a rule set, with rules easily understandable by a human (see Subsection 3.3.5). It could be used in the situation when an explanation or justification of the derived decision is important.

To sum up, the RIONA algorithm combines instance- and rule-based approaches. It uses rules allowing grouping both numerical and symbolic attributes. As a result of using the appropriate size of the neighbourhood of a test case, it is both efficient and effective (in classification). Additionally, searching for the optimal size is also done efficiently. Moreover, RIONA classifiers have the property of explainability.

Chapter 4

RIONIDA

This chapter presents a new learning algorithm, called RIONIDA, which is dedicated to imbalanced data and combines the instance- and rule-based approaches. RIONIDA is the acronym of Rule Induction with Optimal Neighbourhood for Imbalanced Data Algorithm.

The RIONIDA algorithm is based on a modification of the RIONA algorithm. The fundamental idea in developing RIONIDA is to reconstruct RIONA taking into account the issues related to imbalanced data (see Section 2.4). Thus, RIONIDA is an algorithm-level approach to imbalanced data (see Subsection 2.5.2).

To simplify the task, the number of decision classes in RIONIDA is limited to only two (i.e. RIONIDA is directly applicable only for binary classification problems; see the second paragraph in Section 1.3).

The following section introduces the main ideas behind the RIONIDA algorithm. Section 4.2 presents the idea of more general rules in comparison to the ones used in RIONA, which enable to realise one of the ideas of RIONIDA. Section 4.3 describes the whole RIONIDA algorithm as well as its important components (including both ideas coming from RIONA and newly proposed ideas primarily related to imbalanced data). Section 4.4 presents the learning part of RIONIDA, i.e. estimation of the internal parameters of the algorithm. Section 4.5 discusses computational and space complexity of the algorithm along with possibilities of reducing it. Section 4.6 summarises two important properties of the RIONIDA algorithm from the perspective of understanding its process of decision making.

The RIONIDA algorithm, analogously to RIONA, has three parts: initialisation, training and testing. Some comments on the formal structure of the whole RIONIDA algorithm can be found in Subsection 4.4.3.

4.1 Main ideas behind the RIONIDA algorithm

The RIONIDA algorithm, analogously to the RIONA algorithm, is based on a combination of instance-based learning and rule induction. However, in the construction of RIONIDA, some significant changes have been made in comparison with RIONA aiming to develop classifiers for imbalanced data with the highest possible quality. We present here the summary of the changes. They are described in more detail in the following sections.

First, RIONIDA performs optimisation during the learning phase relative to a performance measure more relevant for imbalanced data (e.g. F-measure or G-mean).

Second, the minority class is treated in a special way during the conflict resolution. Another problem is related to choosing to what extent the minority class is more important than the majority class.

Third, because sometimes the ONN algorithm is competitive with the RIONA algorithm, we decided to combine the power of both of them. One can decide whether to use rules in the neighbourhood or not. Moreover, one can define a possibility of a 'smooth' transition between the rule-based approach and the instance-based approach. Thus, we can set a degree to which using rules in the neighbourhoods is important. If this degree drops below zero the pure kNN method is used. If this degree is equal to 1, the pure rule-based approach is used (still restricted in the neighbourhood). The degree between 0 and 1 corresponds to a combination of the instance- and rule-based approaches.

Fourth, all the previously mentioned features of the RIONIDA algorithm, as well as aspects of the RIONA algorithm (adapted to RIONIDA), are automatically induced during the learning phase. It is important to note that we present an efficient in time methods of learning all of these factors by the dynamic programming technique.

4.2 Extension of generalised local decision rule

In this section, we present the idea of the *scaled generalised local decision rule*. It is a further extension of the generalised local decision rule (g-rule) used for the RIONA algorithm (see Section 3.2, Definition 3.2). The idea is to select between rule-based method (as the RIONA algorithm does) or kNN method (as the ONN algorithm does), and, on the other hand, to allow a smooth transition between these approaches.

Definition 4.1. *For any test object tst and any training object trn, we define the* scaled generalised local decision rule *(for short, the* sg-rule*), denoted by sg-rule* $\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}, s\right)$ *or simply sg-rule* (tst, trn, s) *(whenever parameters* $\{\varrho_a\}_{a\in A_{sym}}$ *are clear from the context or irrelevant due to generality), the decision rule with the decision* $d(trn)$ *and the following conditions* t_a *for each attribute* a*:*

$$t_a = \begin{cases} a \in [a(tst), a(tst) + l \cdot s] & \text{when } s \geq 0, a \text{ is numerical, } a(tst) \leq a(trn) \\ a \in [a(tst) - l \cdot s, a(tst)] & \text{when } s \geq 0, a \text{ is numerical, } a(tst) > a(trn) \\ a \in B\left(a(tst), r_a \cdot s\right) & \text{when } s \geq 0, a \text{ is symbolic} \\ a \in \emptyset & \text{when } s < 0, \end{cases}$$

where $l = |a(tst) - a(trn)|$, $r_a = \varrho_a(a(tst), a(trn))$, *and* $B(c, R)$ *is the closed pseudometric ball of radius* R *centred at point* c *for pseudometric* ϱ_a, $s \in [-1, 1]$ *is the scaling parameter of the rule.*

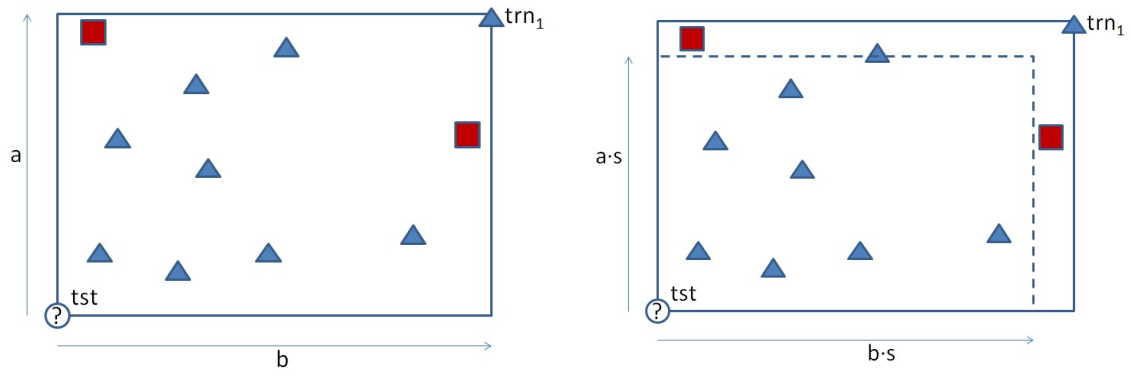
The sg-rule is constructed in such a way that it always contains the test example, and the interval or ball corresponding to each attribute is scaled by the given parameter s in comparison to the original g-rule. It should be observed that for $s = 1$, this definition is equivalent to Definition 3.2. For $s = 0$ we have the rule

covering only the test example and the training examples identical with the test example for all numerical attributes and distanced by 0 for all symbolic attributes. For $s < 0$, the premise of this rule is always false (formally speaking, not satisfied by any example) what relates to elimination of consistency checking and in consequence to working as the kNN algorithm. The parameter s such that $0 < s < 1$ defines the scaling of the satisfiability area of the rule.

We illustrate the idea of the sg-rules with two examples corresponding to numerical and symbolic attributes.

Figure 4.1 presents two sg-rules for two different values of the parameter s in the case of a data set with two numerical attributes. Figure 4.1(a) illustrates the area of satisfiability of sg-rule for $s = 1$ (equivalent to the standard g-rule). Figure 4.1(b) presents the area of satisfiability of sg-rule for some $s < 1$. The first rule is inconsistent with the training data set, while the second is consistent.

Analogously, Figure 4.2 presents two sg-rules for two different values of the parameter s in the case of a data set with one symbolic attribute. The first one is inconsistent with the training data set, while the second is consistent.



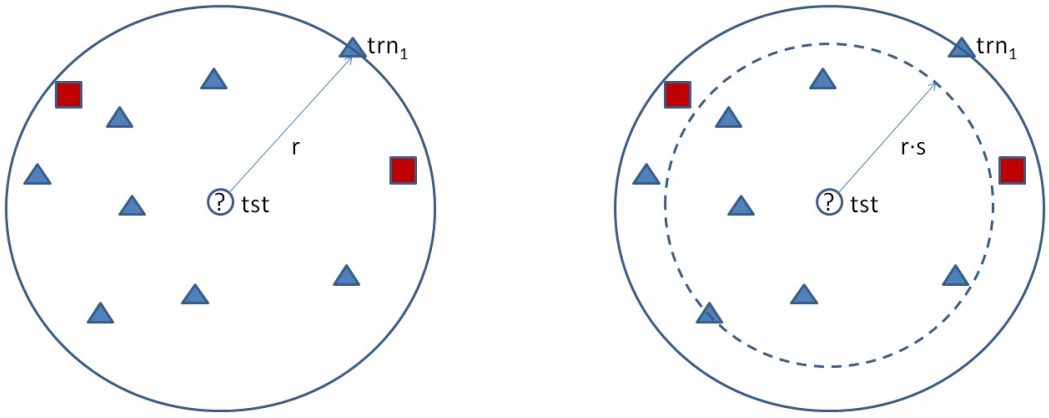
(a) inconsistent sg-rule for $s = 1$

(b) consistent sg-rule for some $s < 1$

Figure 4.1: The sg-rule with decision 'triangle' constructed for a test object *tst* (with an unknown decision) depicted with a circle and a training object trn_1 with decision 'triangle'. The example is for the data set with two numerical attributes. The difference between the values of first and second attribute for a test and train examples over which the local rule is built on is equal to a and b, respectively. Setting $s < 1$ scales the lengths of all intervals with value of s in comparison to the original g-rule. The sg-rules shown in examples are (a) inconsistent for $s = 1$ because there exists an object labelled by a square in the area of sg-rule (b) consistent for some $s < 1$ due to the fact that in the smaller area of satisfiability (of dashed rectangle) there is no square satisfying the sg-rule (all objects labelled by squares are 'outside' of the area of satisfiability of the sg-rule)[1].

[1]It should be noted that objects in examples are represented by points from R^2 with coordintes defined by values of the considered two attributes.

[2]It should be noted that geometry of objects in examples is defined by a given pseudometric for the considered symbolic attribute.



(a) inconsistent sg-rule for $s = 1$ **(b)** consistent sg-rule for some $s < 1$

Figure 4.2: The sg-rule with decision 'triangle' constructed for a test object (with an unknown decision) depicted with a circle and a training object trn_1 with decision 'triangle' for different values of the parameter s. The example concerns the data set with one symbolic attribute. The difference between values of the symbolic attribute for test and train examples the sg-rule is built on is equal to r. Any change of the parameter s leads to scaling of the area of satisfiability of the rule. The sg-rules shown in examples are (a) inconsistent for $s = 1$ because there exists a square in the area of sg-rule (b) consistent for some $s < 1$ due to smaller area of satisfiability of the rule (dashed circle) so that no square satisfy the sg-rule (all squares are 'outside' of the area of satisfiability of the sg-rule)[2].

4.3 RIONIDA description

The RIONIDA algorithm is an extension of the RIONA algorithm for imbalanced data. We already have mentioned some possible extensions of the RIONA algorithm in Section 3.7. Those already mentioned and other significant changes in RIONA are made when constructing the RIONIDA algorithm focused on imbalanced data. Compared to the RIONA algorithm, the following changes have been made:

- adding the possibility of choosing of the performance measure to be optimised – in fact, performance measures dedicated to imbalanced data are taken into account,
- setting sensitivity constraint (for the minority class) to a higher level; furthermore, this sensitivity is adjusted to the currently analysed data,
- providing not only a possibility to learn when to use the kNN method and when rule-based method, but also a combination of both types of algorithms (by tuning levels of rules inconsistency provided in Section 4.2 a smooth transition between both types of algorithms is incorporated).

It should be noted that in our approach after choosing the performance measure (which is relevant to the user needs), the learning phase is performed relative to this chosen performance measure. In consequence, in our approach the same chosen performance measure is used both in training and testing. The internal parameters (size of the neighbourhood – parameter k, sensitivity to the minority class – parameter

p, the degree to which the rules are used – parameter s) are learned during the learning phase.

The RIONIDA algorithm, like any learning algorithm, consists of two parts: learning and classifying (testing). At the step of learning, an essential element of the algorithm is to learn the optimal parameters discussed above. By observing that the space of possible parameters defines a set of many possible classifiers, the task of learning is transformed to the selection of one specific classifier from the given space of classifiers.

For each parameter, there is a set of values that we take into account. The sets of admissible values for the parameters k, p, s, we denote by K, P and S, respectively. Thus, the set of possible classes of classifiers that we search for are of the cardinality $|K| \cdot |P| \cdot |S|$. It should be noted that the space of all possible classifiers is determined not only by these parameters but also by the training set (analogously to kNN method when a distinct classifier is defined for each training set).

The learning process of these optimal parameters is discussed in Section 4.4. Algorithm 9 presents the RIONIDA algorithm for the testing phase. In the following sections, we discuss and analyse in more detail the purpose of using these parameters and other components of the RIONIDA algorithm. It should be pointed out that analogously to RIONA, in the argument of the algorithm all pseudometrics are given (used for computation of the final pseudometric), but in the sg-rule only pseudometrics for symbolic attributes are used (see Definition 3.2 and note after it).

This algorithm is a modification of Algorithm 5 (RIONA). The differences between the RIONIDA and RIONA algorithms are as follows.

- Instead of selecting the class with the highest support, one of the two classes is chosen: the minority class if the support of this class is above a certain threshold (parameter p) or the majority class in the other case.
- Instead of g-rules, sg-rules depending on the parameter s are used; this is related to checking whether the rule is consistent to some extent.

Technically speaking, these differences correspond respectively to the following differences in the algorithm:

- instead of line 13 in Algorithm 5, we have lines 12-17 in Algorithm 9,
- instead of g-rule in line 9 in Algorithm 5, we have sg-rule in line 8 in Algorithm 9.

One should observe that at the step of classification, a performance measure dedicated to imbalanced data does not appear in the RIONIDA algorithm. Here, it is assumed that these parameters have been optimised for this chosen (by a user) measure. However, in the analysis below, we will refer to the relevant performance measures for imbalanced data.

It should be noted that the RIONA algorithm is obtained from RIONIDA after setting the threshold p at 0.5, the parameter s at 1.0, and the optimisation measure to Accuracy. The ONN algorithm is obtained after setting the threshold p at 0.5, the parameter s at -0.1, and the optimisation measure to Accuracy. Thus, RIONIDA is an extension of RIONA and ONN as well.

The analysis presented below aims to show that it is reasonable to introduce the discussed additional parameters and to search the space of them at the learning step. We want to show that different settings of these parameters, may bring significant differences in the values of the performance measures for the imbalanced data. In order to show the importance of these parameters, we will make some simplifications of the original Algorithm 9. It should be noted that the following analysis was prepared after performing the experiments reported in Chapter 5. The analysis justifies both the introduced modifications of the RIONA algorithm and the quality of the experimental results (presented in Chapter 5).

The following issues related to the RIONIDA algorithm are analysed below:

- selection of performance measure for optimisation (in Subsection 4.3.1),
- choice of the neighbourhood size (in Subsection 4.3.2),
- choice of the sensitivity of the minority class (in Subsections 4.3.3-4.3.4),
- choice of the sg-rule factor (in Subsection 4.3.5).

Algorithm 9: RIONIDA-classify(tst, $trnSet$, k, p, s, $\{\varrho_a\}_{a\in A}$)

Input: test example tst, training set $trnSet$, positive integer k, number $p \in [0,1]$, number $s \in [-1,1]$, family of pseudometrics for attributes $\{\varrho_a\}_{a\in A}$

Output: predicted decision $d \in \{d_{min}, d_{maj}\}$ for tst

begin
  $\varrho = Agr(\{\varrho_a\}_{a\in A})$
  $neighbourSet = N(tst, trnSet, k, \varrho)$
  $supportSet(d_{min}) = \emptyset$
  $supportSet(d_{maj}) = \emptyset$
  foreach $trn \in neighbourSet$ **do**
    $v = d(trn)$
    if $isConsistent\left(sg\text{-}rule\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}, s\right), neighbourSet\right)$ **then**
      $supportSet(v) = supportSet(v) \cup \{trn\}$
    end
  end
  $p_{current} = \frac{|supportSet(d_{min})|}{|neighbourSet|}$
  if $p_{current} \geq p$ **then**
    return d_{min} **else**
      return d_{maj}
    end
  end
end

4.3.1 Selection of performance measure for optimisation

Generally, at some step of the data mining process, one establishes a performance measure expressing the quality of a classifier. Normally, this measure is used during the evaluation of the learning algorithm at the testing stage. However, it is natural to make use of this measure and optimise it at the learning stage. In fact, we make use of it in the development of the RIONIDA algorithm.

In the RIONA algorithm, the Accuracy was used to evaluate this algorithm, and this performance measure was estimated at the learning stage. In the RIONIDA algorithm, performance measures, more relevant for imbalanced data, e.g. F-measure, G-mean, AUC or others, could be used. Currently, in the algorithm, one can choose F-measure, G-mean or Accuracy at the learning stage. Primarily, we choose one out of two: F-measure or G-mean (performance measures relevant for imbalanced data).

It should be noted that an essential feature of the RIONIDA algorithm is a possibility to fix a performance measure for which we want to optimise the algorithm. This measure is fixed explicitly, in contrast to many algorithms that implicitly perform some optimisation and then are tested for a certain set of performance measures. The RIONIDA algorithm could also be easily modified so that one could choose any performance measure based on the confusion matrix. For example, it could be a combination of F-measure and G-mean measures.

4.3.2 Choice of the neighbourhood size

While discussing the RIONA algorithm, we observed that the values of the Accuracy measure could drastically change after changing the neighbourhood size. Moreover, we noticed that the optimal size of the neighbourhood could be bounded by a small number, e.g. 200. For the RIONIDA algorithm, we took 100 as a default bound for k. In the performed experiments, we show that taking the bound 200 for k does not change the results significantly (see Subsubsection 'Different maximal k value' on page 208).

Here, as it was mentioned in the previous section, we are interested in performance measures more relevant for imbalanced data, i.e. F-measure or G-mean. One can ask if we obtain similar differences in quality for these measures depending on the size of the neighbourhood. It turns out that we do. Taking only this factor in the modification of the RIONA algorithm to imbalanced data can lead to significant improvement in the quality of imbalanced data classification.

To construct a simplified version of the RIONIDA algorithm, let us try to set the default values of the parameters p and s. A natural candidate for the default value of the parameter p is the percentage of the minority class in the training set, i.e. $\frac{|Class(d_{min})|}{|trnSet|}$ (see also Subsection 4.3.4 with some theoretical argument for this selection). As the default value of parameter s, let us take $s = -0.1$ for which the sg-rule works exactly as for the nearest neighbours (see Section 4.2). In this way, we get an algorithm that we call ONIDA (Optimal Neighbourhood for Imbalanced Data Algorithm). Algorithm 10 presents the ONIDA algorithm. This algorithm can be seen as an extension of the ONN algorithm for imbalanced data.

From now on, we will present estimations of the classification quality (relative to F-measure or G-mean) depending on some parameters of the RIONIDA algorithm (or

ONIDA, the particular case of RIONIDA). The classification quality (in the function of parameters) was computed using the leave-one-out method applied to the whole data set.

First, we discuss how parameter k affects the optimisation of performance measure for the RIONIDA algorithm. For clarity of presentation, we first present estimation for the ONIDA algorithm, simplified version of the RIONIDA algorithm with two default parameters fixed as it was mentioned above. Let us also assume that we want to optimise G-mean measure. In the following figures, we present these dependencies.

Figure 4.3 shows the dependency of G-mean measure on the parameter k for *glass* data set (for details about this data set and others mentioned below see Subsection 5.1.3). For this data set, it can be observed that while increasing k beyond a certain small value (around 10) the G-mean measure is systematically falling down. It is clear from that graph that using a different setting of value k can produce classifiers with completely different quality. In the graph, we observe differences in G-mean of about 40%.

Figure 4.4 shows the same dependency for *breast-cancer* data set (on two different scales). From the graph in Figure 4.4(a), one can see that the maximal value for G-mean is for k higher than 20. Additionally, from that k value, G-mean seems to be stable with respect to changes of values of the parameter k. Moreover, it can be seen that the differences for different values of k can reach about 15%.

Let us look more deeply for this graph using a more relevant scale. In Figure 4.4(b), it is possible to see that the region of the maximal value for this measure can be found for the value of the parameter k around 50. The differences of G-mean for different k bigger than 20 can be about 5%. Again, it shows that searching (during the learning phase) for the appropriate size of the neighbourhood (k) can improve the performance of the algorithm in terms of the chosen performance measure.

Similar results were observed in the performed experiments for F-measure, i.e. choosing the proper size of the neighbourhood (parameter k) can improve the classification quality for the ONIDA algorithm, the simplified version of the RIONIDA algorithm.

4.3.3 Balancing Sensitivity and Specificity

In the case of the RIONA algorithm, we assumed that the cardinality of decision classes is fairly evenly distributed. Conflict resolution was done in favour of the most-represented class in the neighbourhood of the test object.

In the case of imbalanced data, we assume that the minority class may be under-represented. To increase the chance of correct classification of objects from the minority class, objects from this class should be treated differently in comparison to those from the majority class.

For this purpose, we introduce a parameter p used to define how important the minority class is. This is, in a sense, an analogous to changes made in the MODLEM-C algorithm (in comparison to MODLEM), where a fixed weight for examples from the minority class is assigned.

The parameter p of the RIONIDA algorithm determines the minimum rate value

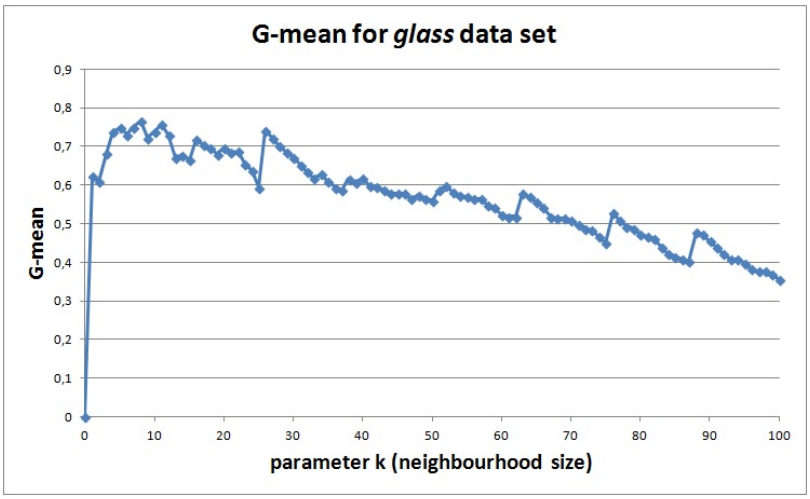
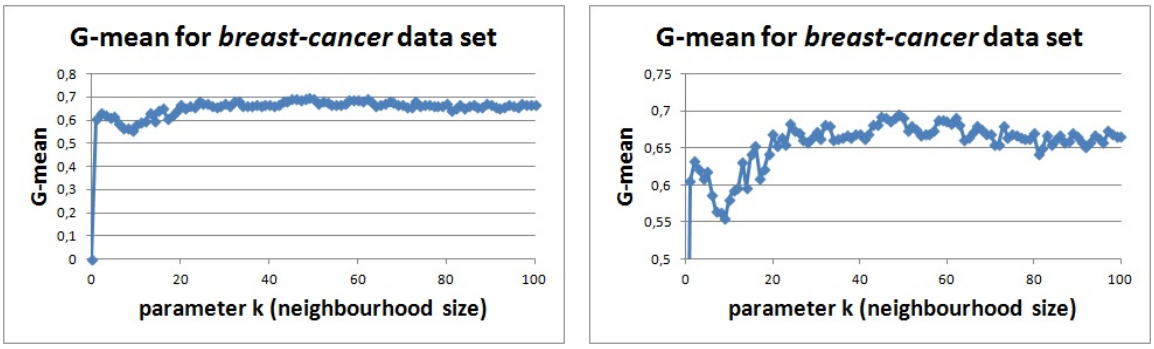


Figure 4.3: G-mean measure for *glass* data set for the ONIDA algorithm as a function of parameter k (neighbourhood size).



(a) normal scale **(b)** scale from 0.5

Figure 4.4: G-mean measure for *breast-cancer* data set for the ONIDA algorithm (for two different scalings: (a) normal and (b) from 0.5) as a function of parameter k (neighbourhood size).

Algorithm 10: ONIDA(tst, $trnSet$, k, $\{\varrho_a\}_{a\in A}$)

Input: test example tst, training set $trnSet$, positive integer k, $\{\varrho_a\}_{a\in A}$ – family of pseudometrics for attributes

Output: a decision $d \in \{d_{min}, d_{maj}\}$

```
1  begin
2      ϱ = Agr({ϱ_a}_{a∈A})
3      neighbourSet = N(tst, trnSet, k, ϱ)
4      p = |Class(d_min)| / |trnSet|
5      supportSet(d_min) = ∅
6      supportSet(d_maj) = ∅
7      foreach trn ∈ neighbourSet do
8          v = d(trn)
9          supportSet(v) = supportSet(v) ∪ {trn}
10     end
11     p_current = |supportSet(d_min)| / |neighbourSet|
12     if p_current ≥ p then
13         return d_min else
14             return d_maj
15         end
16     end
17 end
```

of the number of objects (forming consistent sg-rules) from the minority class to the size of the whole neighbourhood for assigning the minority decision to the test object tst. For example, the value $p = 0.1$ indicates that it is enough to find 10% of objects from the minority class among the nearest objects to tst so that the minority decision is assigned to it. The value of $p = 0.5$ corresponds to the majority decision strategy as it was done in the RIONA algorithm. In this sense, it is another extension of the RIONA algorithm. Of course, we assume that the minority class is more important than the majority class with respect to correct classification. Therefore, in the carried out experiments, in the P set (of admissible values of the parameter p during the learning process) usually only values less than 0.5 are considered.

We can rewrite the condition in line 13 of Algorithm 9 as follows.

$$\begin{aligned}
&\frac{|supportSet(d_{min})|}{|neighbourSet|} \geq p \Leftrightarrow \\
&|supportSet(d_{min})| \geq p \cdot (|supportSet(d_{min})| + |supportSet(d_{maj})|) \Leftrightarrow \\
&(1-p) \cdot |supportSet(d_{min})| \geq p \cdot |supportSet(d_{maj})| \,.
\end{aligned}$$

The above equivalences imply that the condition in line 13 is equivalent to assigning weights to minority and majority examples with values $1 - p$ and p, respectively. In the case of $s = -0.1$, it can be treated as the kNN method with the relevant weights assigned depending on the class to which the object belongs.

Let us look at this more deeply. One could ask a question: What is the meaning of the parameter p for performance measures we try to optimise? Different values of this parameter give different weights for the minority class and the majority class. In consequence, this is related to different levels of sensitivity to the minority class i.e. Sensitivity (and inversely, sensitivity to the majority class, i.e. Specificity). Hence, one could use graphs in the ROC space. Analogously Precision-Recall space can be informative for such analysis.

From the perspective of optimisation of G-mean measure relative to p, it is important to search for a harmonious balance between Sensitivity and Specificity. The ROC curve presents how the change in Specificity affects Sensitivity. The exemplary graph for *yeast* data set is presented in Figure 4.5. One can see from this graph that the optimal value of G-mean measure is obtained by selecting the point of the ROC curve closest to the (0,1) point. This point corresponds to the situation where both Sensitivity and Specificity are balanced and are relatively high.

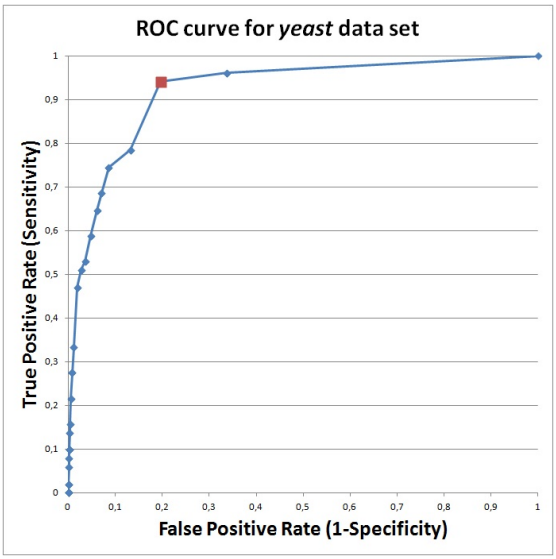


Figure 4.5: ROC graph for *yeast* data set for $k = 48$. Different points in this graph correspond to different values of p (these points are connected by straight lines). The red and bigger point on the graph corresponds to the optimal value of the G-mean measure which is obtained for $p = 0.03$.

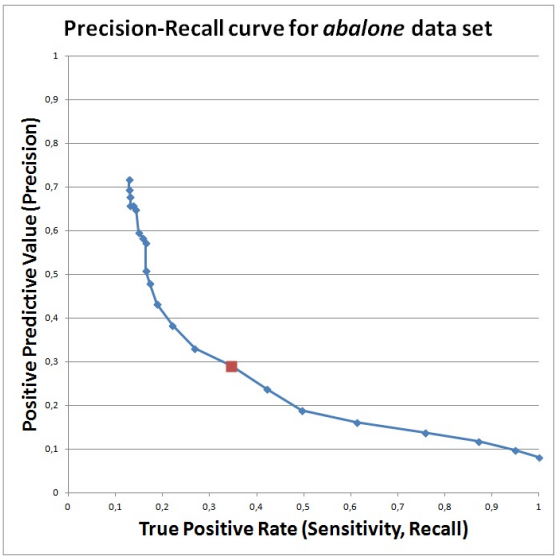


Figure 4.6: Precision-Recall curve for *abalone* data set for $k = 41$. Different points in this graph correspond to different values of $p \in \{0, 0.5\}$ (these points are connected by straight lines). The red and bigger point on the graph corresponds to the optimal value of the F-measure which is obtained for $p = 0.15$.

From the perspective of the optimisation of F-measure relative to p, it is important to obtain a harmonious balance between Sensitivity and Precision. The Precision-Recall curve presents how the change of Sensitivity (Recall) affects Precision. The exemplary graph for *abalone* data set is presented in Figure 4.6. One can see from this graph that the optimal value of F-measure is obtained by selecting the point of the Precision-Recall curve closest to the (1,1) point. That point relates to the situation where both Sensitivity (Recall) and Precision are balanced as well as are relatively high.

We described an idea how the changes of values of the parameter p influence the considered performance measures (G-mean and F-measure). Now, we want to show how changes of values of the parameter p can affect the optimised measure under different values of k. In other words, we want to observe how different pairs of the parameters p and k can affect values of the performance measure we are interested in. In order to limit ourselves to these two parameters, we need to simplify the RIONIDA algorithm. Thus, we assume that the parameter s is fixed at -0.1, which corresponds to the kNN method with variable size of the neighbourhood (parameter k) and variable weights of the minority and majority classes (parameter p). In such a case, we can present a 2-dimensional surface showing the dependency of G-mean on the parameters k and p. It will show the dependency of the quality of the RIONIDA on the two parameters k and p. Figure 4.7 shows the dependency of G-mean measure on both parameters k and p (for an exemplary data set). Figure 4.8 shows the same surface rotated by 130 degrees. These two figures can give a kind of insight into how the G-mean may depend on these two parameters.

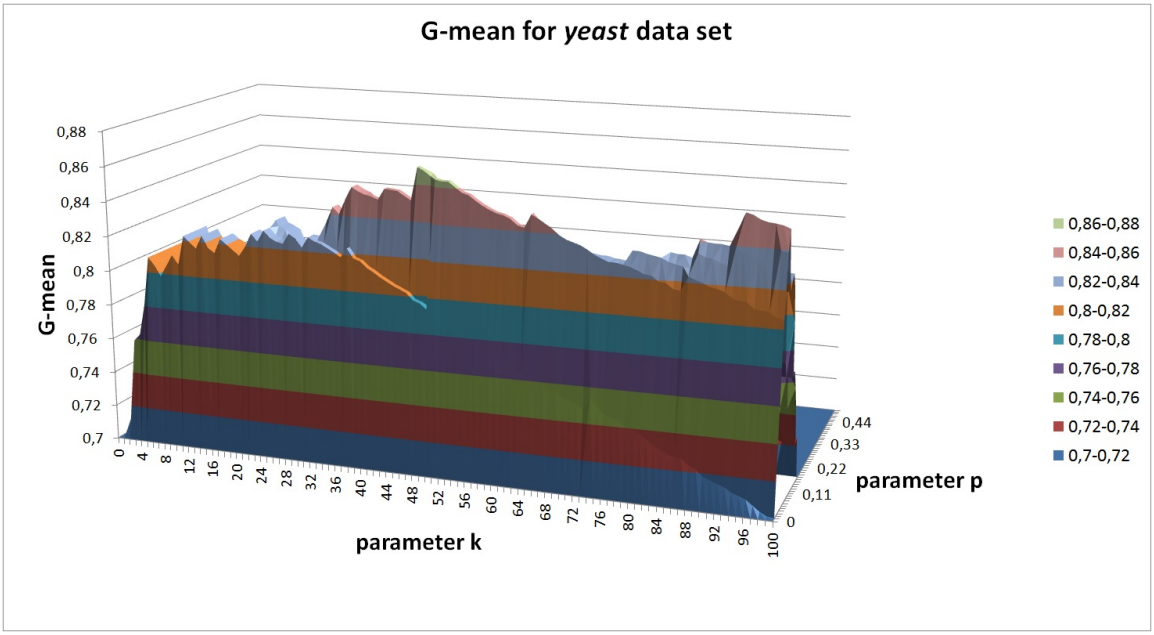


Figure 4.7: Surface chart representing G-mean measure (scaled from 0.7) for the RIONIDA algorithm for *yeast* data set as a function of parameters k and p with fixed $s = -0.1$.

Let us present the cutoff for the surface at $k = 48$ for which it reaches the maximum value (among the parameters k and p). Figure 4.9 presents the dependency of G-mean for the RIONIDA algorithm for constants $k = 48$ and $s = -0.1$. It is visible that the maximum G-mean value is obtained for $p = 0.03$ and $p = 0.04$. At the same time, it can be seen that the ridge of the surface presented in Figures 4.7 and 4.8 runs around these values $p = 0.03$ and $p = 0.04$. Let us note that the percentage of the minority class in *yeast* data set equals 3.44%, which is close to 0.03 (and 0.04). It is consistent with the intuition given before. This issue is discussed below (see Theorem 4.1 and comments following this theorem).

However, the maximum value for the performance measure under consideration can be reached for different values of the parameter p than the value of p equal to

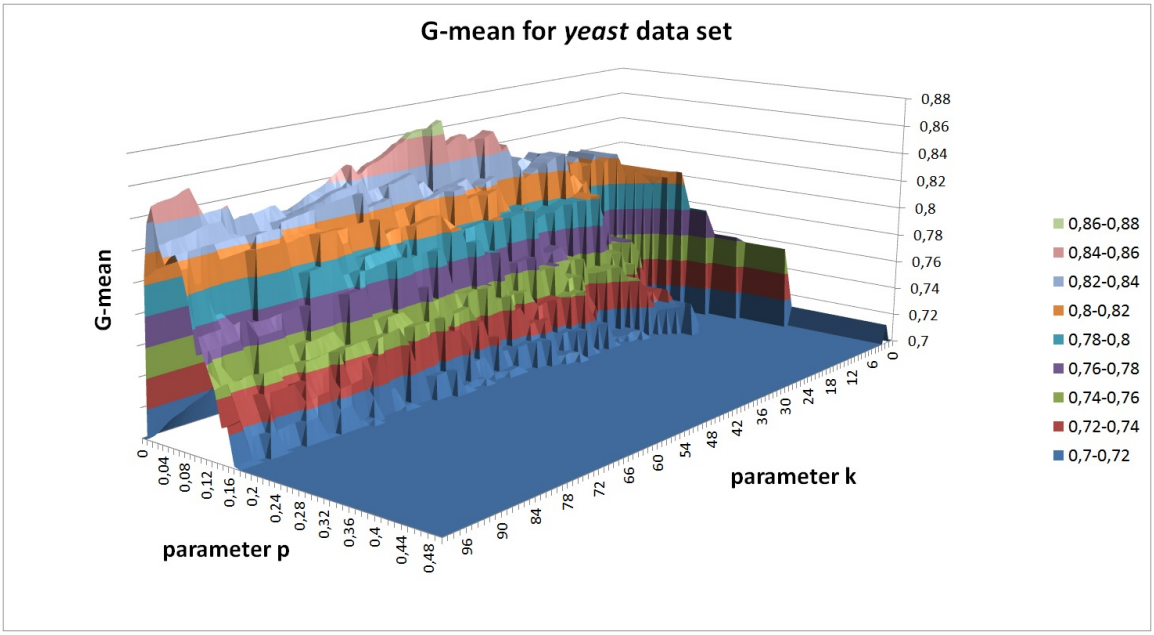


Figure 4.8: Surface chart representing G-mean measure (scaled from 0.7, rotated by 130 degrees) for the RIONIDA algorithm for *yeast* data set as a function of parameters k and p with fixed $s = -0.1$.

the percentage of the minority class. Let us present such an example.

Figure 4.10 shows the dependency of F-measure on both parameters k and p. Let us note that Figure 4.10 for F-measure significantly differs from Figure 4.7 for G-mean measure[3]. The visible maximal points (and points close to the maximal) for these two surfaces are in completely different areas. Moreover, the ridge of both surfaces runs for different values of p.

Let us also present the cutoff for this surface at $k = 12$ for which it reaches the maximum value. Figure 4.11 presents the dependency of F-measure for the RIONIDA algorithm for constants $k = 12$ and $s = -0.1$. It is visible that the maximum F-measure value is obtained for $p \in [0.26, 0.33]$. These values are relatively far from the percentage of the minority class equal to 0.03 value.

All these considerations are supporting a hypothesis that it is worth to find the optimal value of the parameter p according to the given performance measure we want to optimise. This optimal value of the parameter p can be different for different performance measures.

4.3.4 Default candidate for parameter p

The default candidate for the parameter p of RIONIDA in the case of G-mean is the percentage of the size of the minority class from the size of the whole data set. Theorem 4.1 together with the discussion that follows it can be treated as an intuitive explanation of why this default choice can be really good.

In *safe regions* (i.e. regions containing only safe examples) it is easy to classify examples (see Subsections 2.4.2 and 2.4.3). However, in regions of borderline

[3]The fact that the first figure is scaled does not play significant role in what we argue here.

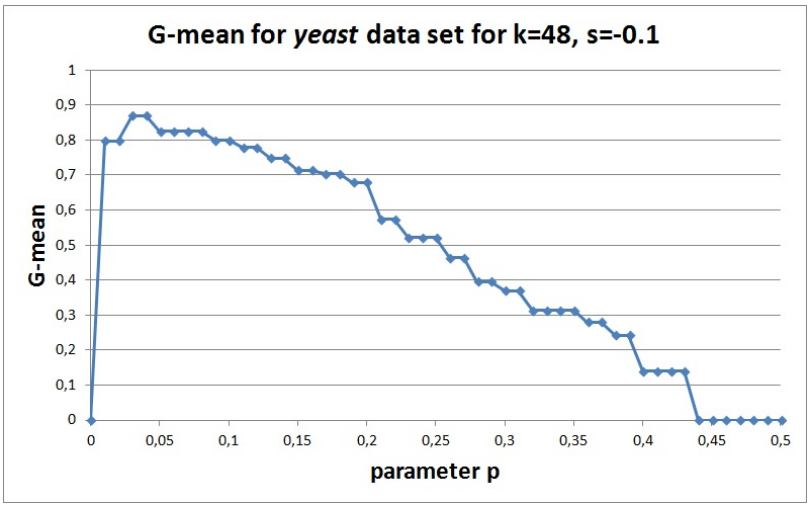


Figure 4.9: G-mean measure for the RIONIDA algorithm for *yeast* data set as a function of parameter p with fixed $k = 48$ and $s = -0.1$.

examples where the majority and minority classes are mixed, the classification becomes harder. We restrict our considerations to the case consisting of borderline examples only which additionally are 'totally mixed' (with fixed imbalance ratio). If we find the optimal parameter for this case, it should also be relevant for safe regions.

For the purpose of this subsection only (and Appendix C with Fact C.1), some useful conventions and notations are presented below.

The decision function, considered here, can be non-deterministic. It means that training examples can be inconsistent.

We decompose $\mathbf{X}$ into the space of vectors of values of conditional attributes, X, and the space of values of the decision attribute d, V_d, i.e. $\mathbf{X} = X \times V_d$. Also, we consider the binary set of decisions $V_d = \{d_{maj}, d_{min}\}$, where $d_{maj} = 0$ and $d_{min} = 1$. Any *example* (*object*) is a pair $(x, d) \in X \times V_d$. Any training set $trnSet$ of the length n is a sequence of examples, i.e. $trnSet = (z_1, \ldots, z_n)$, where $z_i \in X \times V_d$ for $i = 1, \ldots, n$.

By $z \sim \mathcal{D}$ we denote random sampling of z from a set Z according to $\mathcal{D}$, where $\mathcal{D}$ is a probability distribution over Z. Usually, we denote by $\mathcal{D}$ a probability distribution over the set $\mathbf{X} = X \times V_d$. By $trnSet \sim \mathcal{D}^n$ we denote random sampling of the training set $trnSet$ of size n, where each example from $trnSet$ is sampled independently using the same distribution $\mathcal{D}$.

By $\mathbf{E}R$ we denote the expected value of the given random variable R. The subscript of $\mathbf{E}$ in $\mathbf{E}_{z\sim\mathcal{D}}R(z)$ is used to indicate that sampling of z is according to the probability distribution $\mathcal{D}$. Analogously, we denote by $\Pr_{z\sim\mathcal{D}}(Event(z))$ the probability of the event $Event$, where sampling of z is according to the probability distribution $\mathcal{D}$.

The Accuracy of a given classifier C is equal to the probability that this classifier correctly classifies any test example (see e.g. [149]), i.e. $Acc(C) = \Pr_{(x,d)\sim\mathcal{D}}(C(x) = d) = \mathbf{E}_{(x,d)\sim\mathcal{D}}(I(C(x) = d))$, where $C(x)$ is the decision assigned to x by the classifier C, d is the correct decision on x and $I(\cdot)$ is the indicator function (equal to 1, if the

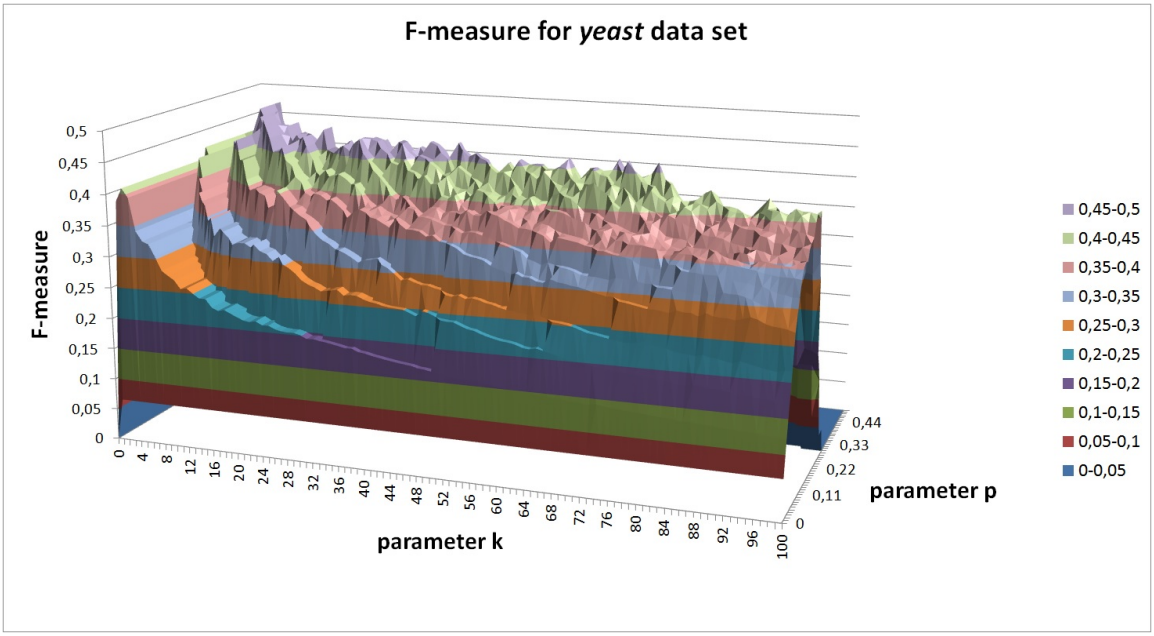


Figure 4.10: Surface chart representing F-measure for the RIONIDA algorithm for *yeast* data set as a function of parameters k and p with fixed $s = -0.1$.

condition in the argument is satisfied and 0, otherwise)[4]. For calculating G-mean we need Sensitivity (called also Accuracy for Positive Class or Recall) and Specificity (called also Accuracy for Negative class):

$$Acc_{min}(C) = \Pr_{(x,d)\sim\mathcal{D}}(C(x) = d \mid d = d_{min}),$$
$$Acc_{maj}(C) = \Pr_{(x,d)\sim\mathcal{D}}(C(x) = d \mid d = d_{maj}).$$

For calculating F-measure, we need Sensitivity and Precision. Precision is the conditional probability that the classification is correct provided that the classifier predicts the positive (minority) class:

$$Prec(C) = \Pr_{(x,d)\sim\mathcal{D}}(C(x) = d \mid C(x) = d_{min}).$$

However, we are interested in computing Accuracy of a learning algorithm $Alg(trnSet)$ constructing a classifier from a given training set $trnSet$ (see Section 2.1). Formally, Accuracy should be averaged over all possible training data sets of fixed size n (see e.g. [149]), i.e. we need to calculate $AvgAcc(Alg) = \mathbf{E}_{trnSet\sim\mathcal{D}^n} Acc(Alg(trnSet)) = \mathbf{E}_{trnSet\sim\mathcal{D}^n} \Pr_{(x,d)\sim\mathcal{D}}(Alg(trnSet)(x) = d) = \mathbf{E}_{trnSet\sim\mathcal{D}^n}\mathbf{E}_{(x,d)\sim\mathcal{D}} I(Alg(trnSet)(x) = d)$[5]. Analogously, for calculating measures related to G-mean and F-measure we need the measures presented above averaged

[4] $I(C(x) = d) = 1 - L(C(x), d)$, where L is the 0-1 loss function (equal to 0, if C correctly classifies the given example (x, d) and 1, otherwise).

[5] Formally, $\Pr_{(x,d)\sim\mathcal{D}}(Alg(trnSet)(x) = d)$ is a random variable, where $trnSet$ is fixed. It can also be seen as the conditional probability on $\mathcal{D}^n \times \mathcal{D}$ given a training set $trnSet$, i.e. $\Pr_{trnSet\sim\mathcal{D}^n,(x,d)\sim\mathcal{D}}(Alg(trnSet)(x) = d \mid trnSet)$.

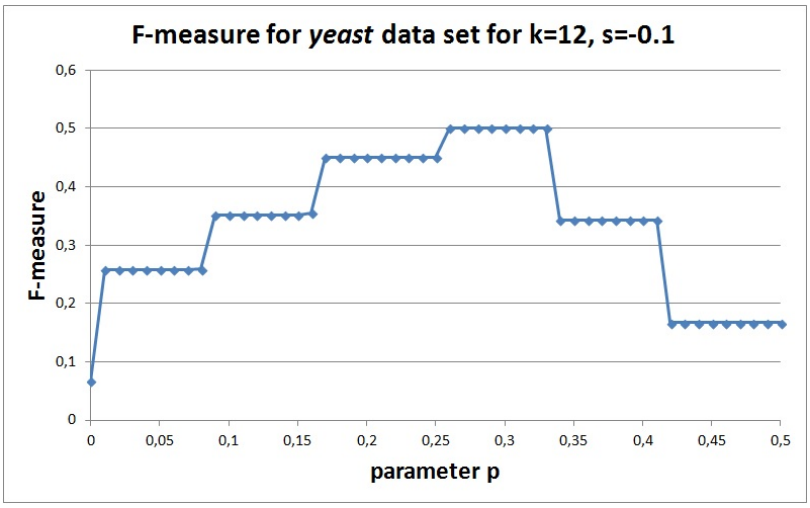


Figure 4.11: F-measure for the RIONIDA algorithm for *yeast* data set as a function of parameter p with fixed $k = 12$ and $s = -0.1$.

over all possible training data sets of fixed size n. Hence, we introduce:

$$\begin{aligned} AvgAcc_{min}(Alg) &= \mathbf{E}_{trnSet\sim\mathcal{D}^n} Acc_{min}(Alg(trnSet)), \\ AvgAcc_{maj}(Alg) &= \mathbf{E}_{trnSet\sim\mathcal{D}^n} Acc_{maj}(Alg(trnSet)), \\ AvgPrec(Alg) &= \mathbf{E}_{trnSet\sim\mathcal{D}^n} Prec(Alg(trnSet)). \end{aligned}$$

For each test example $tst = (x, d)$, any sequence of training examples $trnSet = ((x_1, d_1), \ldots, (x_n, d_n))$, and any pseudometric ϱ, let $\pi_1(x), \ldots, \pi_n(x)$ be the permutation of $\{1, \ldots, n\}$ reordering $(x_1, \ldots, x_n)$ according to $\varrho(x, x_i)$, as follows

$$\varrho(x, x_{\pi_i(x)}) \leq \varrho(x, x_{\pi_{i+1}(x)}), \text{ for each } i \in \{1, \ldots, n-1\}.$$

Without loss of generality for our considerations, one can assume that the permutation is determined uniquely. It should be noted that ϱ may depend on $trnSet$ (see, e.g. $SVDM$ pseudometric in Subsection 2.2.2).

As it was mentioned at the beginning of this subsection, we consider the 'totally random' distribution over set $\mathbf{X}$. Intuitively, it means that for this distribution the decisions of examples for the majority and minority classes are 'totally mixed' (with fixed imbalance ratio) without any dependence on values of conditional attributes. Formally, this means that the distribution $\mathcal{D}$ over $\mathbf{X}$ can be expressed as the product of independent distributions $\mathcal{D}_{\mathcal{X}}$ and $\mathcal{D}_{\mathcal{V}}$ over X and V_d, respectively, i.e. $\mathcal{D} = \mathcal{D}_{\mathcal{X}} \times \mathcal{D}_{\mathcal{V}}$.

In our considerations, Alg is roughly interpreted as the RIONIDA learning algorithm for the fixed k and $s = -0.1$. It is parametrised by $p \in [0, 1]$. Hence, $AvgAcc_{min}$, $AvgAcc_{maj}$ and $AvgPrec$ are functions of p.

Now, we present a theorem which roughly says that the optimal value for the parameter p in the case of G-mean for the RIONIDA algorithm under the assumption of the 'totally random' distribution is very close to the percentage of the size of the minority class from the size of the whole data set.

Theorem 4.1. ***(version for G-mean)*** *Let $k, n \in \mathbb{N}$, $k \leq n$, $q \in (0, 1)$ be given constants. Let $p \in [0, 1]$ be a parameter. Let $\mathcal{D}$ be a distribution over $\mathbf{X} = X \times V_d$ such*

that $\mathcal{D} = \mathcal{D}_{\mathcal{X}} \times \mathcal{D}_{\mathcal{V}}$, *where* $\mathcal{D}_{\mathcal{X}}$ *in any distribution over* X *and* $\mathcal{D}_{\mathcal{V}}$ *is the Bernoulli*(q) *distribution taking values* $d_{min} = 1$ *with probability* q *and* $d_{maj} = 0$ *with probability* $1-q$. *Let* $tst = (x, d) \sim \mathcal{D}$, $trnSet = ((x_1, d_1), \ldots, (x_n, d_n)) \sim \mathcal{D}^n$. *Let* D_i *be a random variable equal to* $d_{\pi_i(x)}$, *i.e. the decision of the* i*-th nearest neighbour (from* $trnSet$*) to* x. *Let us consider the random variable* Alg *with arguments* $trnSet$ *and* x *taking the decision on the basis of values* $D_1(trnSet, x), D_2(trnSet, x), \ldots, D_k(trnSet, x)$ *defined as follows*

$$Alg(trnSet)(x) = \begin{cases} d_{min} & \text{if } \frac{1}{k}\sum_{i=1}^{k} D_i(trnSet, x) > p, \\ d_{maj} & \text{otherwise.} \end{cases}$$

Let us consider the function $AvgGmean(p) = \sqrt{AvgAcc_{min}(p) \cdot AvgAcc_{maj}(p)}$.

If we consider all the values p_{opt} *such that the function* $AvgGmean(p)$ *takes the maximal value at* p_{opt}, *then*

$$\inf_{p_{opt}} |p_{opt} - q| \leq \frac{\ln 2}{k}.$$

Proof. For any fixed $trnSet$ we have

$$\begin{aligned} Acc_{min}(Alg(trnSet)) &= \Pr_{(x,d)\sim\mathcal{D}}(Alg(trnSet)(x) = d \mid d = d_{min}) \\ &= \Pr_{(x,d)\sim\mathcal{D}}(Alg(trnSet)(x) = d_{min} \mid d = d_{min}) \\ &= \Pr_{(x,d)\sim\mathcal{D}}(Alg(trnSet)(x) = d_{min}) \qquad (4.1) \\ &= \Pr_{x\sim\mathcal{D}_{\mathcal{X}}}(Alg(trnSet)(x) = d_{min}) \qquad (4.2) \end{aligned}$$

Equation 4.1 follows from the fact that events $Alg(trnSet)(x) = d_{min}$ and $d = d_{min}$ are independent. Equation 4.2 follows from the fact that $\mathcal{D} = \mathcal{D}_{\mathcal{X}} \times \mathcal{D}_{\mathcal{V}}$.

Analogously, we have

$$Acc_{maj}(Alg(trnSet)) = \Pr_{x\sim\mathcal{D}_{\mathcal{X}}}(Alg(trnSet)(x) = d_{maj}).$$

For any $trnSet$ we have

$$\begin{aligned} &Acc_{min}(Alg(trnSet)) + Acc_{maj}(Alg(trnSet)) = \\ &\qquad P_{x\sim\mathcal{D}_{\mathcal{X}}}(Alg(trnSet)(x) = d_{min}) + P_{x\sim\mathcal{D}_{\mathcal{X}}}(Alg(trnSet)(x) = d_{maj}) = 1. \end{aligned}$$

Hence, we also have

$$\begin{aligned} &AvgAcc_{min}(Alg) + AvgAcc_{maj}(Alg) = \\ &\qquad \mathbf{E}_{trnSet\sim\mathcal{D}^n} Acc_{min}(Alg(trnSet)) + \mathbf{E}_{trnSet\sim\mathcal{D}^n} Acc_{maj}(Alg(trnSet)) = 1. \end{aligned}$$

Thus, we have

$$AvgGmean(p) = \sqrt{AvgAcc_{maj}(p) \cdot (1 - AvgAcc_{maj}(p))}.$$

The root square function is monotonic and under the root square we have a quadratic function of $AvgAcc_{maj}$, which achieves the maximal value for $AvgAcc_{maj} = \frac{1}{2}$. This

quadratic function is symmetrical (around $\frac{1}{2}$) and monotonically increasing up to $AvgAcc_{maj} = \frac{1}{2}$ and from that point monotonically decreasing. Thus, $AvgGmean(p)$ achieves the maximal value for any p_{opt} such that:

$$p_{opt} \in \arg\min_{p\in[0,1]} |AvgAcc_{maj}(p) - \frac{1}{2}|.$$

It should be noted that the above formula not defines the optimal value of p uniquely; we obtain the set of optimal values of p. We consider all such optimal values p_{opt}. Later on we prove that the set of all optimal values p_{opt} is 'close' to the value q.

We have

$$\begin{aligned}
AvgAcc_{maj}(Alg) &= \mathbf{E}_{trnSet\sim\mathcal{D}^n} Acc_{maj}(Alg(trnSet)) \\
&= \mathbf{E}_{trnSet\sim\mathcal{D}^n} \Pr_{x\sim\mathcal{D}_\mathcal{X}} (Alg(trnSet)(x) = d_{maj}) \\
&= \mathbf{E}_{trnSet\sim\mathcal{D}^n} \mathbf{E}_{x\sim\mathcal{D}_\mathcal{X}} I(Alg(trnSet)(x) = d_{maj}) \qquad (4.3) \\
&= \mathbf{E}_{trnSet\sim\mathcal{D}^n} \mathbf{E}_{(x,d)\sim\mathcal{D}} I(Alg(trnSet)(x) = d_{maj}) \\
&= \mathbf{E}_{trnSet\sim\mathcal{D}^n,(x,d)\sim\mathcal{D}} I(Alg(trnSet)(x) = d_{maj}) \\
&= \Pr_{trnSet\sim\mathcal{D}^n,(x,d)\sim\mathcal{D}} (Alg(trnSet)(x) = d_{maj}) \qquad (4.4) \\
&= \Pr_{trnSet\sim\mathcal{D}^n,(x,d)\sim\mathcal{D}} (\sum_{i=1}^{k} D_i \leq pk) \qquad (4.5) \\
&= F_{B(k,q)}(pk), \qquad (4.6)
\end{aligned}$$

where $B(k,q)$ denotes the binomial distribution and $F_{B(k,q)}(v)$ its cumulative distribution function at point v, i.e. $F_{B(k,q)}(v) = \sum_{i=0}^{\lfloor v \rfloor} \binom{k}{i} q^i (1-q)^{(k-i)}$, $\lfloor v \rfloor$ is the 'floor' under v, i.e. the greatest integer less than or equal to v.

Equations 4.3 and 4.4 follow from the definition of indicator function. Equation 4.5 follows from the definition of Alg. Any permutation of training examples (formally, random variables) does not change their distribution and independence, thus for any $1 \leq i \leq k$, $D_i \sim Bernoulli(q)$ (D_i are identically distributed) and D_i are (mutually) independent. Thus, the probability in Equation 4.5 is the cumulative distribution function of binomial distribution $B(k,q)$ at point pk. This implies Equation 4.6.

From the previous considerations we obtain the set of all optimal values p_{opt} which satisfy:

$$p_{opt} \in \arg\min_{p\in[0,1]} |F_{B(k,q)}(pk) - \frac{1}{2}|.$$

Let us denote by $\tilde{p}_{opt}$ the smallest optimal value p_{opt}. Then $\tilde{p}_{opt}k$ is an integer value since the cumulative binomial distribution function is a step function with jumps in integer values and constant between them. First, let us consider a specific case when the cumulative distribution function achieves the optimal value (i.e. the closest to $\frac{1}{2}$) at both integer values $\tilde{p}_{opt}k$ and $\tilde{p}_{opt}k + 1$. Then all the optimal values p_{opt} form the interval $[\tilde{p}_{opt}, \tilde{p}_{opt} + \frac{2}{k})$ since the optimal values p_{opt} are contained in the sum of two intervals for which $F_{B(k,q)}$ is closest to $\frac{1}{2}$, i.e. $[\tilde{p}_{opt}, \tilde{p}_{opt} + \frac{1}{k})$ and $[\tilde{p}_{opt} + \frac{1}{k}, \tilde{p}_{opt} + \frac{2}{k})$. We will return to this case at the end of the proof. From now on, we consider the opposite case. Then all the optimal values p_{opt} form the interval $[\tilde{p}_{opt}, \tilde{p}_{opt} + \frac{1}{k})$.

For the need of the considerations that follow, it is worthwhile to recall the definition of the median. The median of the distribution induced by a random variable R is any real number m that satisfies the inequalities:

$$\Pr(R \leq m) \geq \frac{1}{2} \text{ and } \Pr(R \geq m) \geq \frac{1}{2}.$$

In the three cases considered below it will be convenient to denote by e the value $\tilde{p}_{opt}k = \lfloor \tilde{p}_{opt}k \rfloor$ and by D the random variable with distribution $B(k,q)$.

The first case: the optimal (i.e. the closest to $\frac{1}{2}$) value $F_{B(k,q)}(\tilde{p}_{opt}k)$ is equal to $\frac{1}{2}$. Then $P(D \leq e) = F_{B(k,q)}(e) = \frac{1}{2}$; $P(D \geq e) > P(D > e) = 1 - P(D \leq e) = \frac{1}{2}$. Hence, e is the median of $B(k,q)$.

The second case: the optimal value $F_{B(k,q)}(\tilde{p}_{opt}k)$ is less than $\frac{1}{2}$. Then $P(D \leq e) = F_{B(k,q)}(e) < \frac{1}{2}$. Then $P(D \leq e+1) = F_{B(k,q)}(e+1) > \frac{1}{2}$ (otherwise $F_{B(k,q)}(e)$ would not be the optimal value). We also have $Pr(D > e) = 1 - P(D \leq e) > \frac{1}{2}$. Thus, $Pr(D \geq e+1) = Pr(D > e) > \frac{1}{2}$. It implies that $e+1$ is the median of $B(k,q)$.

The third case: the (optimal) value $F_{B(k,q)}(\tilde{p}_{opt}k)$ is greater than $\frac{1}{2}$. Then $P(D \leq e) = F_{B(k,q)}(e) > \frac{1}{2}$. We also have $P(D < e) = P(D \leq e-1) < \frac{1}{2}$ (otherwise $F_{B(k,q)}(e)$ would not be the optimal value). Then $P(D \geq e) = 1 - P(D < e) > \frac{1}{2}$. Hence, that e is the median of $B(k,q)$.

To sum up, we have shown that for all $\tilde{p}_{opt}$, we have that $e = \tilde{p}_{opt}k$ or $e+1$ is the median of $B(k,q)$.

On the other hand in [96] (see also [115]) it is shown that any median M of $B(k,q)$ cannot be 'far' from its mean value $\mu = kq$. More precisely, the distance between M and μ can be at most $\ln 2$, i.e.:

$$|M - \mu| \leq \ln 2. \tag{4.7}$$

This implies that for any $\tilde{p}_{opt}$ we have (respectively for the cases when e is the median or $e+1$ is the median)

$$|e - kq| \leq \ln 2 \text{ or } |e + 1 - kq| \leq \ln 2.$$

Thus,

$$|\tilde{p}_{opt}k - kq| \leq \ln 2 \text{ or } |\tilde{p}_{opt}k + 1 - kq| \leq \ln 2.$$

Hence,

$$|\tilde{p}_{opt} - q| \leq \frac{\ln 2}{k} \text{ or } |\tilde{p}_{opt} + \frac{1}{k} - q| \leq \frac{\ln 2}{k}. \tag{4.8}$$

Let us recall that all the optimal values p_{opt} form the interval $[\tilde{p}_{opt}, \tilde{p}_{opt} + \frac{1}{k})$. Therefore, either (in the case when e is the median) the beginning of this interval is distanced from q not more than $\frac{\ln 2}{k}$ or (in the case when $e+1$ is the median) the end of it is distanced from q not more than $\frac{\ln 2}{k}$. Thus,

$$\inf_{p_{opt}} |p_{opt} - q| \leq \frac{\ln 2}{k}.$$

We still have to prove the theorem for the specific case when all the optimal values p_{opt} form the interval $[\tilde{p}_{opt}, \tilde{p}_{opt} + \frac{2}{k})$. Then from the above considerations, it is easy to see that in such case the value q belongs to this interval. Hence, it belongs to the set of optimal values p_{opt}.

□

We used in the proof of the above theorem the best possible approximation between the median and the mean (independent of q and k) of the binomial distribution (see [96]; see also [115]). However, it should be noted that we do not want to look for the maximal distance between median and mean, but the distance between q and the interval of optimal values p_{opt}. In particular, in many cases this interval contains the value q.

It seems that for $q < \frac{1}{2}$ it is possible to find a much better bound on the distance between q and this interval using Fact C.1 formulated, proved and shortly commented in Appendix C.

Intuitively, the algorithm Alg in Theorem 4.1 represents the RIONIDA algorithm with a fixed parameter k (given in the assumption of the theorem), $s = -0.1$, and with a data set represented only by a single region with the high degree of overlapping between classes, i.e. only borderline examples occur in the data set under consideration (see Subsections 2.4.2 and 2.4.3). For technical reasons, we only require that Alg takes d_{maj} (instead of d_{min} as the RIONIDA algorithm does) in the situation when $\frac{1}{k}\sum_{i=1}^{k} D_i = p$ (see the formulation of the theorem). The function $AvgGmean(p)$ represents the G-mean for RIONIDA (with fixed parameters as described above) for different values of the parameter p. The theorem roughly says that the maximal G-mean value for RIONIDA is achieved for p equal roughly to q (the percentage of the size of the minority class). For example, without going to the technical details, the theorem says that if there are 5% of examples from the minority class mixed totally randomly with examples from the majority class, then the optimal value for the parameter p for RIONIDA is achieved for $p = 5\%$.

Obviously, in practice, data sets contain not only borderline examples but also safe examples. Thus, it would be valuable for applications to formulate and prove more general theorem for borderline and safe regions. We leave it for future work. However, below we give an intuitive explanation that, roughly speaking, the theorem's conclusions will remain true in such more general situation.

First, let us assume that there are some other regions with borderline examples with the same overlapping level of the minority and majority classes (formally, distributed randomly with the same parameters of Bernoulli distribution). If one adds such regions, the conclusion of the theorem will also hold since it can be treated as one region of borderline examples.

Second, let us consider the case when both borderline and safe examples occur in data. In this case, one can divide the whole space of examples into the safe region (consisting of the safe regions of the majority class and the safe regions of the minority class) and the borderline region (consisting of borderline regions in different areas of data). Let us assume that all examples from the safe region are correctly classified by the algorithm (independently of the parameter p)[6]. Let us also assume that the global percentage of the minority class is the same as the percentage of the minority class in the borderline region. Under these assumptions, it is easy to check that the optimal parameter p will be the same as in the theorem's conclusion (i.e. the optimal parameter p for the borderline region).

This shows that, in a sense, only regions with borderline examples are important

[6]In practice, such an assumption is satisfied for a wide range of values of the parameter p.

to focus on in order to achieve the high G-mean value.

In the case of dealing with real-life data sets, even if borderline examples are 'totally mixed', the given above assumptions may be not satisfied. For instance, some examples treated as safe examples can be misclassified for the optimal value of the parameter p, the borderline regions can have different percentage of the minority class, or the global percentage can be different from the percentage of borderline regions. However, if the given above assumptions are 'roughly' satisfied, then the optimal parameter p can be only 'slightly' different from that given in the theorem.

Concerning the parameter p, RIONIDA, in fact, is searching for the relevant value of the parameter p in case the distribution is not totally random but is slightly directed toward one class (is, in a sense, between the borderline region and safe region of one class). If there exist two regions with borderline examples with different parameters of Bernoulli distribution, then RIONIDA searches for the optimal value of the parameter p treating these regions of borderline examples as one. This observation indicates that for different borderline regions different optimal values of the parameter p should be searched for. This is one of the topics for future work (see Section 6.2).

Now, we present the theorem which roughly says that the optimal value of the parameter p in the case of F-measure for the RIONIDA algorithm and 'totally random' distribution is 0.

Theorem 4.2. ***(version for F-measure)*** *Under the assumptions of Theorem 4.1 let us consider the function* $AvgFmeasure(p) = H(AvgAcc_{min}(p), AvgPrec(p))$, *where* $H(\cdot,\cdot)$ *is the function of harmonic mean of its arguments, i.e.* $H(a,b) = \frac{2}{a^{-1}+b^{-1}}$.

Then the function $AvgFmeasure(p)$ *takes the maximal value at*

$$p_{opt} \in \left[0, \frac{1}{k}\right).$$

Proof. Let us recall that by $F_{B(k,q)}$ we denote the cumulative binomial distribution function.

$$\begin{aligned} AvgAcc_{min}(Alg) &= 1 - AvgAcc_{maj}(Alg) \\ &= 1 - \Pr_{(x,d)\sim\mathcal{D},S\sim\mathcal{D}^m}\left(\sum_{i=1}^{k} D_i \leq pk\right) \\ &= 1 - F_{B(k,q)}(pk). \end{aligned}$$

Both the first and second equation come from the proof of Theorem 4.1. The third equation follows from the fact that the probability in the previous equation is equal to the cumulative distribution function of the binomial distribution $B(k,q)$ at point pk (for details see the proof of Theorem 4.1).

We also have

$$\begin{aligned}
AvgPrec(p) &= \mathbf{E}_{trnSet\sim\mathcal{D}^n}\mathbf{E}_{(x,d)\sim\mathcal{D}}(I(Alg(trnSet)(x)=d) \mid Alg(trnSet)(x)=d_{min}) \\
&= \mathbf{E}_{trnSet\sim\mathcal{D}^n}\mathbf{E}_{(x,d)\sim\mathcal{D}}\left(I(d_{min}=d) \mid Alg(trnSet)(x)=d_{min}\right) \\
&= \mathbf{E}_{trnSet\sim\mathcal{D}^n}\mathbf{E}_{(x,d)\sim\mathcal{D}}I(d_{min}=d) && (4.9)\\
&= \mathbf{E}_{trnSet\sim\mathcal{D}^n}\mathbf{E}_{x\sim\mathcal{D}_\mathcal{X}}\mathbf{E}_{d\sim\mathcal{D}_\mathcal{Y}}I(d_{min}=d) && (4.10)\\
&= \mathbf{E}_{trnSet\sim\mathcal{D}^n}\mathbf{E}_{x\sim\mathcal{D}_\mathcal{X}}\Pr_{d\sim\mathcal{D}_\mathcal{Y}}(d=d_{min}) && (4.11)\\
&= \mathbf{E}_{trnSet\sim\mathcal{D}^n}\mathbf{E}_{x\sim\mathcal{D}_\mathcal{X}}q && (4.12)\\
&= q
\end{aligned}$$

Equation 4.9 follows from the fact that events $d_{min}=d$ and $Alg(trnSet)(x)=d_{min}$ are independent (for any fixed $trnSet$). Equation 4.10 follows from the fact that $\mathcal{D}=\mathcal{D}_\mathcal{X}\times\mathcal{D}_\mathcal{Y}$. Equation 4.11 follows from definition of the indicator function. Equation 4.12 follows from the fact that $\mathcal{D}_\mathcal{Y}$ is the $Bernoulli(q)$ distribution.

Thus, we have

$$AvgFmeasure(p) = H(AvgPrec(p), AvgAcc_{min}(p)) = H(q, 1-F_{B(k,q)}(pk))$$

The first argument of H with respect to p is constant and H is monotonically increasing function of the second argument. Thus, the function $AvgFmeasure(p)$ takes the maximal value at those values of p for which the function $1-F_{B(k,q)}(pk)$ takes the maximal value. Hence, the function $AvgFmeasure(p)$ takes the maximal value at

$$p_{opt} \in \operatorname*{arg\,min}_{p\in[0,1]} F_{B(k,q)}(pk)$$

Every cumulative distribution function is non-decreasing. Thus, the function $AvgFmeasure(p)$ takes the maximal value at p_{opt} such that $F_{B(k,q)}(p_{opt}k)=0$. From definition of $F_{B(k,q)}(pk)=0$ we have

$$\lfloor p_{opt}k \rfloor = 0$$

Thus

$$p_{opt} \in \left[0, \frac{1}{k}\right).$$

□

Intuitively, function $AvgFmeasure(p)$ represents the F-measure for RIONIDA (with fixed parameters as described above) for different values of the parameter p. Intuitively, the theorem says that the maximal value of F-measure for RIONIDA is achieved for p equal roughly to 0. This relates to the algorithm classifying examples to the minority class if at least one minority example occurs in the neighbourhood. This is intuitively clear because for F-measure we need to balance between Precision and Sensitivity. Precision is constant for totally random examples, i.e. in a sense, it does not depend on algorithm. Thus, to maximise the F-measure one needs to maximise Sensitivity. It is done by setting the minimal possible value of the parameter p. This

is related to classifying all objects to the minority class (excluding only the situations such that all neighbours of a given test object are from the majority class).

It can seem strange that the set of optimal values p_{opt} does not depend on the value of q. For example, both for q close to 0 and close to 1 the optimal value does not change. However, it should be observed that F-measure is the harmonic mean of Sensitivity and Precision. Thus, in a sense, this performance measure 'favours' one class, that is the minority class. This measure does not balance between classifying to the minority class and the majority class, but rather between classifying to the minority class and quality of this classification, i.e. Precision. Hence, if Precision is constant (which is the case when classes are 'totally mixed' with the fixed imbalance ratio), then to maximise F-measure one should choose such p that the classifier chooses the minority class as often as possible. In fact, p close to 0 relates to this case. Irrespective of the value of q, it is better to classify examples to the minority class (if it is only possible). This is an intuitive explanation of the above theorem. This theorem and explanation can also be treated as a kind of criticism of the F-measure (at least in situations similar to described in the theorem). However, it is worth pointing out that for practical data sets Precision is not constant. We then have to balance between Precision and Sensitivity.

Moreover, comparing the optimal values for G-mean and F-measure for the case when classes are 'totally mixed' one can see that the optimal values for different performance measures can be very different. In fact, in the described situation we do not optimise the parameter p according to the given data (since as high randomness occurs one can deduce nothing) but to the selected performance measure[7]. In this sense, these theorems illustrate that in some specific situations learning algorithms may rather 'learn' optimisation measure more than useful relations between conditional attributes and decision. One should be aware of that.

Also, these theorems lead to another interesting observation. To be specific, consider 'random' data set with the percentage of the minority class (i.e. the value of q from the assumptions of the theorems) equal to 0.3 and $k = 50$ (size of the considered neighbourhood). Then, these theorems show that for a given data set (in our case, 'random' data set), the optimal classifiers from a given class of classifiers may be significantly different (in respect to classification) depending on the performance measure relative to which the optimal classifier is selected. Moreover, it can be easily calculated (using formulas from the proofs of the theorems) in the considered case that the assessments of these two optimal classifiers are significantly different depending on the performance measure used for the assessment. For one performance measure, the first optimal classifier is much better than the second one; and for another performance measure, vice versa (the second one is much better than the first one). These observations may help to understand that the 'best' classifier selection may strongly depend on the chosen performance measure. Also, it shows that without a precise specification of what particular performance measure we optimise, the 'best classifier' term can be ambiguous or even misleading. It has practical implications for real-life (data mining and) classification tasks.

Analogously as for the previous theorem (for G-mean), it would be more

[7] Of course, it is well known that in data mining process one defines the optimisation measure at some step of data mining process (see e.g. [59]).

relevant for practical applications to formulate and prove more general theorem with borderline and safe regions. Again, we leave it for future work. The given previously intuitive explanation that the theorem for G-mean can be easily generalised for such a case would not work for F-measure. This is due to the fact that safe examples from the majority class could be misclassified for p close to 0 (which is the value close to the optimal values of p from the theorem for F-measure). In consequence, Precision would be not constant (would depend on p). Then, the optimal values of p in such case could be greater than 0 and should be recalculated for the generalised theorem for F-measure.

4.3.5 Choice of scaling factor in the sg-rule

In the RIONA algorithm, we only count those objects from the neighbourhood that support the consistent g-rule (*isConsistent* method). On the other hand, in the ONN algorithm, we take into account all objects from the neighbourhood. Experiments for the RIONA and ONN algorithms have shown that depending on the data set selection sometimes RIONA and sometimes ONN achieves the better quality. Thus, one can try to learn from the training sample which algorithm to apply for a specific data set. Even more, one can also introduce a smooth transition between these two situations. This is done by introducing the parameter s (see Section 4.2).

The value of s corresponds to the degree of consistency rule detection. The value $s = 1$ corresponds to the situation as in the RIONA algorithm. In this sense, RIONIDA is an extension of this algorithm. The value $s = -0.1$ corresponds to the ONN method, i.e. we do not check the consistency of examples. For consistent data sets, the value $s = 0$ also corresponds to the situation when we use the ONN method. Intermediate values, i.e. $0 < s < 1$ correspond to the situations between the ONN algorithm and the rule-based algorithm.

Figure 4.12 shows the dependency of G-mean measure on both parameters k and s for *haberman* data set. We have fixed here parameter $p = 0.22$ (close to the percentage of the minority class in the whole data set; for this value of p the maximum value of G-mean was obtained over the set of values for the parameters k, p, s). It is visible that for almost all values of the parameter k the maximum value of G-mean is obtained for some value of the parameter s between 0 and 1, near 0.5.

Figure 4.13 shows the dependency of G-mean on the parameter s for fixed $k = 96$ and $p = 0.22$. It is visible that the maximum value of G-mean is obtained for $s = 0.5$ and the difference between maximal (for $s = 0.5$) and minimal (for $s = 1.0$, i.e. for rule-based classifier) value is approximately 10%. The difference between maximum G-mean value (for $s = 0.5$) and G-mean value for $s = -0.1$ (i.e. for the ONIDA algorithm, which gives kNN like classifier) is approximately 8%. Hence, it is clear for these examples that neither rule-based nor kNN-based classifier gives the best result. The best result is obtained for classifier which behaves somehow "between" kNN-based classifier and rule-based classifier.

Of course, one might argue that for kNN-based classifier one could find other optimal values for the parameters p and k. The same argument could be given for rule-based classifier. We try to answer the question whether taking this argument into account the parameter s is important, i.e. whether it can lead to an improvement in

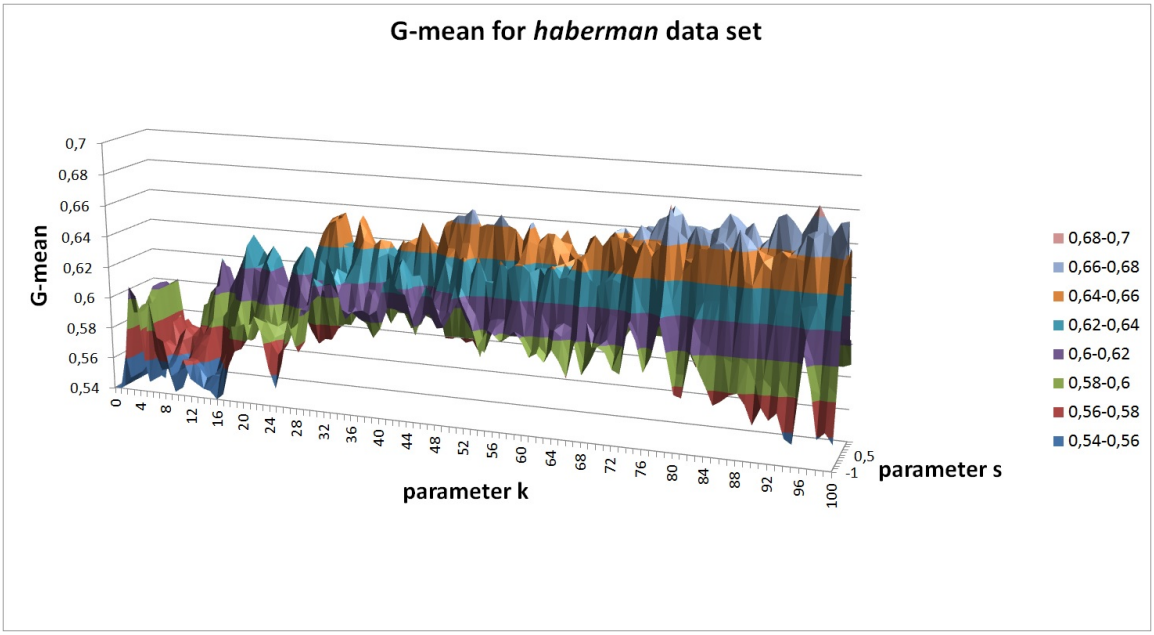


Figure 4.12: Surface chart representing G-mean measure (scaled from 0.58) for the RIONIDA algorithm for *haberman* data set as a function of parameters k and s with fixed $p = 0.22$.

classification. Figure 4.14 shows the dependency of G-mean value on the parameter s under the assumption that for a given parameter s we could find the optimal parameters p and k. We purposely present Figure 4.14 on the same scale as Figure 4.13. Indeed one can see that the graph in Figure 4.14 is flattened in comparison to the graph in Figure 4.13. It means that the previously given differences diminish. However, the differences are still quite significant. The maximum G-mean value in Figure 4.14 is obtained as previously for $s = 0.5$. The difference between maximal and minimal values are approximately 3.2%. The difference between the maximum G-mean value (for $s = 0.5$) and G-mean value for $s = -0.1$ (i.e. for the ONIDA algorithm, which gives kNN like classifier) is approximately 2.1%. The difference between the maximum G-mean value (for $s = 0.5$) and G-mean value for $s = 1$ (i.e. for rule-based classifier) is approximately 1.8%. Still these differences show a possibility for significant improvements of both rule-based classifier and kNN-based classifier by using a classifier 'between' these two.

Generally, we have such a division: for some data sets the maximum value of the optimised measure is reached for $s = -0.1$, i.e. for methods of the kNN type. For another part of data sets, the maximum value is reached for $s = 1.0$, which corresponds to the application of rules. In turn for a part of data sets the maximum value is reached for $s \in (0, 1)$, which corresponds to the application both of these: the rule-based method to some extent and kNN-based method to some extent.

4.3.6 Some specific situations

In this subsection, we present what was done in some specific situations.

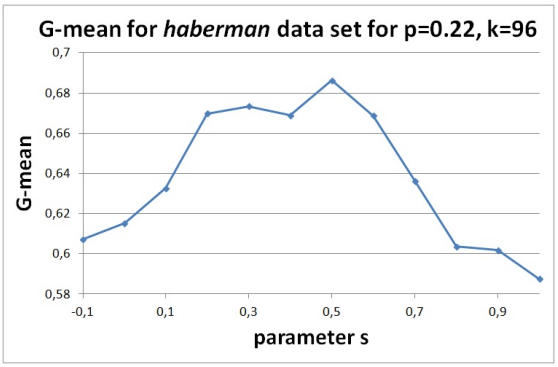


Figure 4.13: G-mean for the RIONIDA algorithm (scaled from 0.58) for *haberman* data set as a function of parameter s with fixed $k = 96$ and $p = 0.22$.

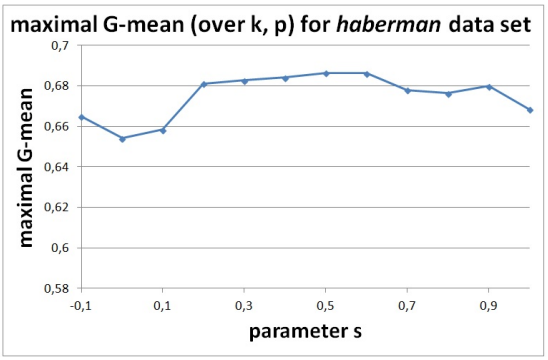


Figure 4.14: Maximal G-mean (scaled from 0.58) over all values of k and p for the RIONIDA algorithm for *haberman* data set as a function of parameter s.

Inconsistencies

RIONIDA, as well as RIONA, works with inconsistent training sets. However, in RIONIDA we use more information than in RIONA in situations when inconsistencies in the training set cause that for a given test example both support sets for decisions are empty.

We describe here how the RIONIDA algorithm works in situations when inconsistencies in data sets occur, i.e. there exist objects undistinguishable by values of conditional attributes but with a different decision. Previously, for clarity, we did not mention this detail in the description of the RIONIDA algorithm.

Let us consider the situation when for a test object presented for classification there exist training objects with the same value for any conditional attribute as the tested example and with different values of the decision attribute. In this case, for all values of the parameter $s \geq 0$, we have $supportSet(d_{min}) = supportSet(d_{maj}) = \emptyset$, i.e. no examples supporting minority or majority decision are found. Therefore, in this situation, it seems a sensible solution to use at least those inconsistent training examples. In this situation, we count the number of inconsistent training examples from each class, and we use these as support sets. We apply this procedure both during learning and classification.

We also accelerate the algorithm for this situation. Before time-consuming examining which training examples form inconsistent rules, we quickly check whether the situation of inconsistencies (described above) occurs. If the situation of inconsistency occurs, then for $s \neq -0.1$ there is no need to investigate which training examples are in the support set, because none of them will be.

There may also be a different situation when the algorithm may return a zero distribution. It may occur if there are training objects close to the test one but with nonzero distance from it, which mutually cause inconsistencies. Then, all sg-rules cover objects with a different decision than that assigned to the rule, and therefore for many levels of s (except levels close to zero) there are no consistent sg-rules. This situation does happen in the examined in experiments data sets, This situation does happen for data sets used in experiments, though very rarely. We do not settle it by any sophisticated way. Naturally, if zero support sets are calculated for both

decisions, then the minority decision (privileged for the classifier) is taken.

The RIONIDA algorithm was also tested for inconsistent data sets. In fact, such data sets occur in the performed experiments (see Subsection 5.1.3).

Missing values

Missing values in the RIONIDA algorithm are treated precisely as in the RIONA algorithm (see Subsection 3.3.1). The RIONIDA algorithm was also tested for data sets with missing values. In fact, such data sets occur in the performed experiments (see Subsection 5.1.3).

4.4 Estimating the optimal values of parameters for RIONIDA

The above considerations (see Subsections 4.3.2, 4.3.3, 4.3.5) show that the performance of RIONIDA can significantly depend on the chosen values of the parameters k, p, s. The optimal values of these parameters depend on the analysed data set and the selected optimisation measure. Therefore, it is essential to find the optimal values of these parameters relative to the optimisation measure specified by a user. It should be noted that the domains of the parameters k, p, s (maximal admissible sets for these parameters) are as follows: $K_{max} = \{1, 2, \ldots, |trnSet|\}$, $P_{max} = [0, 1]$, $S_{max} = \{-0.1\} \cup [0, 1]$. We would like to search for the optimal triple values in the Cartesian product of these sets. From the algorithmic point of view, one should restrict the search to some finite subsets of these sets.

Analogously as in the case of the RIONA algorithm, to construct an efficient algorithm one should take into account the following questions:
(1) For given finite sets K, P, S, how to learn the optimal triple values efficiently from $K \times P \times S$?
(2) Is it possible to select some finite subsets $K \subseteq K_{max}$, $P \subset P_{max}$, $S \subset S_{max}$ of 'small' sizes such that the optimal solution obtained for these sets K, P, S is 'close' to the optimal solution for K_{max}, P_{max}, S_{max}?

4.4.1 Efficient learning of the optimal values of parameters for RIONIDA

In this section, we describe the algorithm for estimation of the optimal values of the parameters k, p, s for the RIONIDA algorithm. This can be done in an analogous way to searching for the optimal value of k in the case of the RIONA algorithm (see Section 3.4). The leave-one-out method is used on the given training set to estimate the value of the performance measure (chosen by a user) for different values of $(k, p, s) \in K \times P \times S$ and the triple values of k, p, s for which the estimation of the measure value is the greatest is selected[8]. The direct calculations require repeating leave-one-out estimation $|K| \cdot |P| \cdot |S|$ times. However, using the dynamic

[8] In the sequel we also denote by k, p, s the values of the parameters k, p, s (treated in algorithms as variables), respectively, if this not leads to confusion.

programming technique, we emulate this process in time comparable to the single leave-one-out test for k equal to the maximal possible value $k = k_{max} = |K|$.

Below we present Algorithm 12 implementing this idea.

For a training example trn the function $getClassificationMatrix$ (see Algorithm 11) finds k_{max} examples from $trnSet \setminus \{trn\}$ nearest to the example trn and sorts them according to the distance $\varrho(trn, \cdot)$ from the trn object.

Next, for any example nn_k from the selected neighbourhood and any $s \in S$, the sg-rule is built on trn (treated as a testing object) and nn_k (treated as a training object), i.e. the rule $sg\text{-}rule\left(trn, nn_k, \{\varrho_a\}_{a \in A_{sym}}, s\right)$. The algorithm checks consistency of this sg-rule with the objects from the neighbourhood for different levels of $s \in S$ and stores this information in the entry corresponding to s of the array assigned to the object nn_k.

Next, it calculates the matrix of decisions that the RIONIDA classifier would return for different triple values $(k, p, s) \in K \times P \times S$ and this matrix is returned as a result.

Algorithm $findOptimalParams3D$ (see Algorithm 12) calls the function $getClassificationMatrix(\dots)$ for every training object. Next, it creates a matrix with the confusion matrix as its entry for each triple $(k, p, s) \in K \times P \times S$. The entry of this matrix corresponding to the index defined by the values of the parameters k, p, s consists of the confusion matrix consisting of information for leave-one-out classification for these values of the parameters k, p, s over all training examples (excluding the considered one). Any confusion matrix (in the matrix of confusion matrices $estimatedConfusionMatrix$) is transformed into one value calculated using the selected optimisation measure $optMeasure$ (and stored in the matrix $estimatedMeasure$). Finally, it selects the triple of the optimal values of the parameters k, p, s for which the global estimation of the chosen optimisation measure is maximal.

This algorithm is analogous to Algorithm 7. In this algorithm, the triple of the optimal values of the parameters k, p, s, rather than only one value of the parameter k is returned. Moreover, the optimal parameters relative to the given optimisation measure instead of the Accuracy measure are returned.

The algorithm $findOptimalParams3D$ has arguments K, P, S specifying the sets of admissible values for the parameters k, p, s, respectively. We assume that $K = \{1, 2, \dots, k_{max}\}$, i.e. the admissible values of the parameter k are consecutive natural numbers.

Another argument of the algorithm is the optimisation measure $optMeasure$. In the current implementation F-measure, G-mean or Accuracy can be substituted here as the value of this argument. However, from the description of the algorithm, it is clear that any optimisation measure, which is the function of the confusion matrix, could also be used.

4.4.2 Bounds on the values of parameters k, p, s

In this subsection, we argue that it is enough to consider sets K, P, S with a small size. This affects the speed of the learning algorithm.

First, we consider the bounds on values of the parameter k. We make use of the

Algorithm 11: getClassificationMatrix(trn, $trnSet$, K, P, S, $\{\varrho_a\}_{a\in A}$)

Input: currently considered example $trn \in trnSet$, training set $trnSet$, K, P, S – sets of admissible values for parameters k, p, s, respectively, family of pseudometrics for attributes $\{\varrho_a\}_{a\in A}$

Output: 3 dimensional matrix (for different triple values $(k, p, s) \in K \times P \times S$) of leave-one-out classification for trn

```
1  begin
2      k_max = |K| (we assume that K is the set of consequent natural numbers)
3      ϱ = Agr({ϱ_a}_{a∈A})
4      N = N(trn, trnSet \ {trn}, k_max, ϱ)
5      vector nn_1, ..., nn_|N| = N sorted according to the distance ϱ(trn, ·)
6      for k = 1 to |N| do
7          for s ∈ S do
8              nn_k.isConsistentOnLevel[s] =
                isConsistent(sg-rule (trn, nn_k, {ϱ_a}_{a∈A_sym}, s), N)
9          end
10     end
11     for s ∈ S do
12         decStrength[d_min] = 0
13         decStrength[d_maj] = 0
14         for k = 1 to |N| do
15             if nn_k.isConsistentOnLevel[s] then
16                 v = d(nn_k)
17                 decStrength[v] = decStrength[v] + 1
18             end
19             p = decStrength[d_min] / (decStrength[d_min]+decStrength[d_maj])
20             for p_current ∈ P do
21                 currentDec = d_min
22                 if p_current > p then
23                     currentDec = d_maj
24                 end
25                 M[k, p, s] = currentDec
26             end
27         end
28     end
29     return M
30 end
```

Algorithm 12: findOptimalParams3D($trnSet$, K, P, S, $optMeasure$, $\{\varrho_a\}_{a\in A}$)

Input: training set $trnSet$,
K, P, S – sets of admissible values for parameters k, p, s, respectively,
optimisation measure $optMeasure$ from
{F-measure, G-mean, Accuracy},
family of pseudometrics for attributes $\{\varrho_a\}_{a\in A}$

Output: triple of the optimal values of parameters k, p, s

```
1  begin
2      foreach trn ∈ trnSet do
3          M_trn = getClassificationMatrix(trn, trnSet, K, P, S, {ϱ_a}_{a∈A})
4      end
5      fill estimatedConfusionMatrix with values 0
6      foreach (k, p, s) ∈ K × P × S do
7          foreach trn ∈ trnSet do
8              realDec = d(trn)
9              classifierDec = M_trn[k, p, s]
10             estimatedConfusionMatrix[k, p, s][realDec, classifierDec] ++
11         end
12         count estimatedMeasure[k, p, s] from
            estimatedConfusionMatrix[k, p, s] based on optMeasure
13     end
14     return  arg max_{(k,p,s)∈K×P×S}  estimatedMeasure[k, p, s]
15 end
```

experience with the RIONA algorithm (see Subsection 3.4.2). Thus, we extend the hypothesis for the RIONIDA algorithm (similarly as it was experimentally checked for RIONA) stating that there is no need to use the whole training set in the process of classification. We also expect, in the case of RIONIDA, that the bound of the neighbourhood size can even improve the classification performance or at least not reduce it significantly. By default, we take $k_{max} = |K| = 100$, which means that $K = K_{def} = \{1, 2, \ldots, 100\}$. We did not perform extensive experiments as in the case of the RIONA algorithm to check the pre-assumed hypothesis. However, for the selected data sets we observed that while increasing k beyond a certain small value the optimisation measure was stable relative to this change or decreasing, i.e. analogously as it was in the case of RIONA and the Accuracy measure (used for RIONA; see Subsection 3.4.2; see also [81]). Moreover, we checked during experiments that setting $k_{max} = |K| = 200$ did not improve the performance of RIONIDA (see Subsubsection 'Different maximal k value' on page 208) what can be treated as an argument for the hypothesis. Also, the promising results of experiments aiming to compare RIONIDA with other algorithms (see Chapter 5) can be treated as an argument for the hypothesis.

Second, we consider a particular set P of admissible values of the parameter p. It should be noted that in the neighbourhood N consisting of k objects, the possible values p in line 19 of Algorithm 11 are of the form $0, \frac{1}{k}, \frac{2}{k} \ldots, \frac{k}{k}$, where $k \leq |N|$. Any two values from the list for a given k fall into two different intervals of the form $[\frac{a-1}{k_{max}}, \frac{a}{k_{max}})$, where $1 \leq a \leq k_{max}$, $a \in \mathbb{N}$ if we only assume that $|N| = k_{max}$. Thus it seems enough to consider the set $P = \{0, \frac{1}{k_{max}}, \frac{2}{k_{max}}, \ldots, 1\}$. Moreover, as we assume that the data sets are imbalanced, therefore we can assume that the minority class is of greater importance than the majority class. In consequence, there is no need to consider values of the parameter p greater than 0.5 (such values indicate for the greater importance of the majority class). Since the selected default value is $|K| = k_{max} = 100$, then by default we take $P = P_{def} = \{0.00, 0.01, 0.02, \ldots 0.5\}$. We checked during experiments also other settings with denser uniform partitions of the interval $[0, 1]$ (see Subsubsection 'Different sets of admissible values for parameter p' on page 208). However, this did not improve the performance of RIONIDA significantly. One can treat this as an argument supporting the claim that the selected kind of partition is sufficient for selecting the optimal value of the parameter p.

Third, we consider the set S of admissible values of the parameter s. In the beginning, we considered $S = S_{def} = \{-0.1, 0.0, 0.1, \ldots, 1.0\}$. This is our default setting. During experiments, we observed that for many data sets there were no differences or small differences (in terms of the optimisation measure value) between two consequent settings of the parameter s. It indicated that there was no need to check the sets S with a larger number of possible values. However, we checked during experiments also other settings for smaller sets S (see Subsubsection 'Different settings of parameter s' on page 208).

In consequence, by default we use sets $K = K_{def}$, $P = P_{def}$, $S = S_{def}$ with a size of 100, 50 and 10 respectively, i.e. with small size.

4.4.3 Comments on the structure of RIONIDA

The general structure of RIONIDA is analogous to the one of RIONA (see Subsection 3.4.3 and Algorithm 8). We present it in Algorithm 14.

The main, initialisation algorithm is analogous to the one in Algorithm 8. Here, we want only to stress that one of the assigned options is the optimisation measure $optMeasure$ (one of F-measure, G-mean, and Accuracy) to be used later during searching for the three optimal internal parameters of RIONIDA. In the main experiments, we did not use Accuracy. However, it can be used as well (see Section 6.2).

The training part (function *RIONIDA-train*) is analogous to the one in Algorithm 8. The difference is in the result that is not the single variable k_{opt} but the triple of variables k_{opt}, p_{opt}, s_{opt}. Moreover, these are searched not according to Accuracy but according to the given option $optMeasure$.

Let us sum up the most important parts of the RIONIDA algorithm shown in Algorithm 14. During initialisation RIONIDA defines Agr, i.e. the aggregations of pseudometrics (by default the sum of pseudometrics for attributes). During training, pseudometrics for attributes are calculated, and the optimal values of the parameters k, p, s (according to $optMeasure$) are searched. These pseudometrics and the optimal values of the parameters k, p, s are used during classification.

Again it should be stressed that both the computation of pseudometrics and the searching for the optimal values k, p, s are always done using only the available training set (e.g. during the cross-validation process). This becomes clear from the description of Algorithm 14.

4.5 Time and space complexity of RIONIDA

In this section, we analyse time and space complexity of RIONIDA. Moreover, we show how the both presented complexity bounds can be improved for the learning phase.

4.5.1 Time complexity of RIONIDA for the testing phase

The analysis of the RIONIDA algorithm in the testing phase is very similar to the RIONA algorithm. In any run of the RIONIDA algorithm, two phases can be distinguished. In the first phase, training examples from the neighbourhood N are selected. In the second phase, the algorithm checks consistency among them. The time complexity of RIONIDA is the same as for the RIONA algorithm. Under the assumption made in Subsection 3.3.2 (on the size of the neighbourhood N), the time complexity of RIONIDA is $O(m(n+k^2))$ for a single test object, where $n = |trnSet|$, $m = |A|$.

Also the same conclusion given in Subsection 3.3.2 for RIONA holds for RIONIDA. Precisely, in the case when k is treated as a constant (or $k < \sqrt{n}$), the time complexity of the testing phase (for single test object) for RIONIDA is $O(mn)$.

4.5.2 Time and space complexity of RIONIDA for the learning phase

Time complexity

The analysis of time complexity of the learning phase for RIONIDA is in many aspects analogous to RIONA (see Subsection 3.4.1). Thus we omit some details already mentioned in Subsection 3.4.1.

Theorem 4.3. *Assume that* $|N| = |N(trn, k_{max})| \leq c \cdot k_{max}$ *for all* $trn \in trnSet$, *where* c *is a constant very close to 1. Then the time complexity of the learning phase of RIONIDA is* $O(mn^2 + n|S| \cdot k_{max} \cdot (mk_{max} + |P|))$, *where* $n = |trnSet|$, $m = |A|$, $k_{max} = |K|$ *is the parameter used to define the maximal size of the neighbourhood to be analysed,* P, S *are sets of admissible values of the parameters* p, s, *respectively (see Section 4.4).*

Proof. For any training object, in the run of the learning algorithm (see lines 2-4 of Algorithm 12) one can distinguish four phases (realised by Algorithm 11).

In the first phase, training examples from the neighbourhood N are selected, i.e. k_{max} nearest objects to the considered training example (or more objects in the specific situation described in Definition 2.14) among n objects, where $n = |trnSet|$. The time complexity of this phase is $O(mn)$, where $m = |A|$ (see Subsection 3.4.1).

In the second phase, all selected objects from the neighbourhood N are sorted. Computing distances for objects from N takes $O(m|N|)$ steps (once for every object from N). Sorting (using computed distances) can be done in $O(|N| \log |N|)$ steps. Thus, this phase takes $O(m|N| + |N| \log |N|)$ steps.

In the third phase, the algorithm checks consistency (and marks it) among selected objects for different values of the parameter s from the set S (see lines 6-10 of Algorithm 11). It takes $O(|S| \cdot m \cdot |N|^2)$ steps.

From the assumption on the bound of the neighbourhood N, the second and third phases altogether take $O(|S| \cdot mk_{max}^2)$ steps.

In the fourth phase, the algorithm fills the classification matrix M on the basis of the marked consistency (see lines 11-28 of Algorithm 11). It takes $|S| \cdot |K| \cdot |P|$ steps.

Thus, the method $getClassificationMatrix$ takes $O(mn + |S| \cdot mk_{max}^2 + |S| \cdot |K| \cdot |P|)) = O(mn + |S| \cdot k_{max}(mk_{max} + |P|))$ steps. This method is executed for each training example. Thus, the time complexity of *foreach* loop within lines 2-4 of Algorithm 12 is $O(mn^2 + n|S| \cdot k_{max} \cdot (mk_{max} + |P|))$.

Finally, for the whole training set, the algorithm computes the leave-one-out confusion matrix for each triple $(k, p, s) \in K \times P \times S$ (see lines 5-13 of Algorithm 12). This takes $O(nk_{max} \cdot |P| \cdot |S|)$ steps.

Summing up, the time complexity of the learning algorithm is $O(mn^2 + n|S| \cdot k_{max} \cdot (mk_{max} + |P|))$. □

If we assume that $|P| \leq mk_{max}$ (which is true in our primary experiments), then the time complexity of the learning algorithm is $O(m(n^2 + n|S| \cdot k_{max}^2))$, where $n = |trnSet|$, $m = |A|$.

Space complexity

Fact 4.4. *The space complexity of the learning phase for RIONIDA is* $O(n \cdot |K| \cdot |P| \cdot |S|)$*, where* $n = |trnSet|$*,* K*,* P*,* S *are sets of admissible values of the parameters* k*,* p*,* s*, respectively (see Section 4.4).*

Proof. The space complexity of the learning phase for RIONIDA is mainly related to allocating matrices for all training examples of the size $|K| \cdot |P| \cdot |S|$ (see lines 2-4 of Algorithm 12). Thus, allocated space is of the size $O(n \cdot |K| \cdot |P| \cdot |S|)$, where $n = |trnSet|$. For the matrices *estimatedConfusionMatrix* and *estimatedMeasure* it is necessary to allocate space $O(|K| \cdot |P| \cdot |S|)$. Thus, the overall space complexity of the learning phase for RIONIDA is $O(n \cdot |K| \cdot |P| \cdot |S|)$. □

4.5.3 Further acceleration of RIONIDA

Estimation of the optimal values of parameters is done efficiently by Algorithm 12 due to dynamic programming used in it. However, it is possible to further accelerate computations performed by this algorithm[9]. Below, we describe how to accelerate the *for* loop in lines 7-9 of Algorithm 11.

The considered *for* loop is inside *for* loop for variable k. Thus, in this section, we assume that the value of variable k, set by *for* loop in line 6 of Algorithm 11, is fixed. In this section, we consider the sg-rule built on trn (treated as a testing object) and nn_k (treated as a training object), i.e. $sg\text{-}rule\left(trn, nn_k, \{\varrho_a\}_{a \in A_{sym}}, s\right)$.

First, it is to be noted that if $isConsistent(sg\text{-}rule\left(trn, nn_k, \{\varrho_a\}_{a \in A_{sym}}, s\right), N)$ is true for some $s = s_0$, then it is also true for all $s \leq s_0$. Analogously, if $isConsistent(sg\text{-}rule\left(trn, nn_k, \{\varrho_a\}_{a \in A_{sym}}, s\right), N)$ is false for some $s = s_0$, then it is also false for all $s \geq s_0$.

Second, it is not necessary to check the consistency for different levels of s as it is done in the *for* loop in lines 7-9 of Algorithm 11. Instead, it is possible to efficiently find an intermediate value s_0 with the following property: $isConsistent(sg\text{-}rule\left(trn, nn_k, \{\varrho_a\}_{a \in A_{sym}}, s\right), N)$ is false for all $s \geq s_0$; and is true for all $s < s_0$. Then such a value s_0 can be used to quickly fill the entries $nn_k.isConsistentOnLevel[s]$ for all $s \in S$. Below we describe how to efficiently calculate the value of s_0 with the described property.

For any $nn_i \in N$ (it is even sufficient to consider smaller number of objects; see remarks in Subsection 3.3.3) such that $d(nn_i) \neq d(nn_k)$, we can calculate the intermediate value $s_1 = s_1(nn_i)$ with the property that $isConsistent(sg\text{-}rule\left(trn, nn_k, \{\varrho_a\}_{a \in A_{sym}}, s\right), \{nn_i\})$ is false for all $s \geq s_1$; and is true for all $s < s_1$. One can simply check consistency of elementary conditions for all attributes. Let us fix an attribute $a \in A$. For $a \in A_{num}$ we can easily calculate the value s for which the value $a(nn_i)$ is on the border of the scaled interval of the sg-rule, e.g. for $a(nn_k) \geq a(trn)$, $s = \frac{a(nn_i) - a(trn)}{a(nn_k) - a(trn)}$ (easily calculated using Definition 4.1). For $a \in A_{sym}$ we can easily calculate the value s for which the value $a(nn_i)$ is on the border of the scaled ball of the sg-rule. The final value of s_1 is chosen as the maximal s for all attributes (for $s < s_1$ at least one elementary condition of sg-rule is not

[9] It should be noted that the considered acceleration obtained during analysis of the algorithm is not yet implemented.

satisfied for object nn_i, hence the whole condition of sg-rule is also not satisfied for it, thus object nn_i cannot cause inconsistency of the rule).

We select s_0 as the minimal value s_1 (obtained for single object as was described above) among all objects from N (for $s \geq s_0$ at least one training object breaks consistency of the rule). For this value of s_0 and all $s \geq s_0$ we have that $isConsistent(sg\text{-}rule\left(trn, nn_k, \{\varrho_a\}_{a\in A_{sym}}, s\right), N)$ is false. For all $s < s_0$ $isConsistent(sg\text{-}rule\left(trn, nn_k, \{\varrho_a\}_{a\in A_{sym}}, s\right), N)$ is true.

The accelerated version of Algorithm 11 is presented in Algorithm 13 (the beginning of the algorithm and its accelerated part is presented only; the dots in line 22 of Algorithm 13 should be replaced by lines 11-28 from Algorithm 11).

Time complexity of the accelerated learning phase of RIONIDA

Additionally, we add an assumption that $|S| < mk_{max}$ (which is true for parameters used in our experiments).

Theorem 4.5. *Under the assumption of Theorem 4.3 and $|S| < mk_{max}$ we have what follows. The accelerated version of the learning phase of the RIONIDA algorithm presented in this subsection (instead of Algorithm 11 is used Algorithm 13) has time complexity $O(mn^2 + nk_{max} \cdot (mk_{max} + |S| \cdot |P|))$, where $n = |trnSet|$, $m = |A|$, $k_{max} = |K|$ is the parameter used to define the maximal size of the neighbourhood to be analysed, P, S are sets of admissible values of the parameters p and s, respectively (see Section 4.4).*

Proof. Computation of the value s_0 with the above-described method requires examination of all attributes on the examples $nn_1, nn_2, \ldots, nn_{k-1}$. This takes $O(km)$ operations. It is done for all k $(1 \leq k \leq |N|)$. Thus, the *for* loop takes $O(mk_{max}^2)$ operations. For such computed s_0 there should also be filled entries $nn_k.isConsistentOnLevel[s]$ for all $1 \leq k \leq |N|$ and all $s \in S$. It takes $O(k_{max} \cdot |S|)$. To sum up, we need to perform $O(k_{max}(mk_{max} + |S|))$ operations.

Using the assumption that $|S| < mk_{max}$, the time complexity is $O(mk_{max}^2)$. Let us note that the time complexity related to lines 6-10 of Algorithm 11 is $O(|S| \cdot mk_{max}^2)$.

If we rewrite the analysis of time complexity from Subsection 4.5.2 with the described acceleration we obtain the time complexity of the method $getClassificationMatrix$

$$O(mn + mk_{max}^2 + |S| \cdot |K| \cdot |P|)) = O(mn + k_{max}(mk_{max} + |S| \cdot |P|)).$$

Thus, the time complexity of *foreach* loop within lines 2-4 of Algorithm 12 and also the time complexity of the learning algorithm is reduced from $O(mn^2 + nk_{max} \cdot (|S| \cdot mk_{max} + |S| \cdot |P|))$ to $O(mn^2 + nk_{max} \cdot (mk_{max} + |S| \cdot |P|))$. □

The significant acceleration could be achieved if the first factor mn^2 is dominated by the others. This can happen for n such that k_{max} is of the size of order $\sqrt{n}$. However, it should be noted that in practice this first factor related to searching for the nearest neighbours has much lower (average) time complexity. In the current implementation, this is achieved by using special data structures for fast searching for the nearest neighbours (see Subsection 3.6.1).

Now, let us assume that the first factor is dominated by the others. In this case, the significant acceleration would be achieved if $|S| \cdot mk_{max} > |S| \cdot |P| \Leftrightarrow mk_{max} > |P|$. The bigger difference of these factors (mk_{max} and $|P|$) we have, the more significant acceleration is achieved (maximally close to $|S|$ times). If these factors are equal, then the acceleration would be no more than two times (and depends on $|S|$).

To sum up, the degree of presented acceleration of the learning phase of RIONIDA mainly depends on the fact whether the first factor responsible for searching of nearest neighbours is dominated by the other factors or not. The acceleration of the remaining factors can be even close to $|S|$ times (depending on the value $\frac{mk_{max}}{|P|}$).

Reduction of the RIONIDA space complexity

Fact 4.6. *The space complexity of the accelerated learning phase for RIONIDA can be reduced to $O(n \cdot |K| \cdot |P|)$, where $n = |trnSet|$, K, P, S are sets of admissible values of the parameters k, p, s, respectively (see Section 4.4).*

Proof. One can observe by taking into account the remarks presented at the beginning of this subsection, that the procedure *findOptimalParams3D* does not have to fill the matrix for all possible values from S. It is sufficient to keep in memory the value s_0 (calculated as described above in this subsection). In consequence, it is enough to remember in memory the matrices of the size $O(|K| \cdot |P|)$ for all training examples. In consequence, the overall space complexity of the learning phase for RIONIDA can be decreased to $O(n \cdot |K| \cdot |P|)$. □

4.6 Important aspects of RIONIDA

The RIONIDA algorithm has two substantial properties from the perspective of understanding its process of decision making. First, in many cases, its behaviour can be interpreted in a way easily understandable by a human. Second, the performance measure, which RIONIDA optimises, is given explicitly. These two topics are discussed below.

4.6.1 Interpretation of the behaviour of RIONIDA

The RIONIDA algorithm, analogously to the RIONA algorithm is based on a combination of instance- and rule-based methods. Moreover, the RIONIDA algorithm is equipped with some parameters which can be tuned in the learning process. In particular, by tuning these parameters, one can obtain the algorithm close in behaviour to one or another of the mentioned classification approaches.

For RIONA there was presented an idea of interpreting its parameters in such a way that RIONA becomes equivalent to rule-based classifier easily understandable by a human (see Subsection 3.3.5). For RIONIDA the situation is more compound. In this case, it can be more difficult or even impossible to interpret in this way all its parameters. However, we present below an idea of the interpretation of parameters of RIONIDA in a way easily understandable by a human.

We assume here that values of the parameters k, p and s in the RIONIDA algorithm are fixed (possibly learned as described in Section 4.4). If an explanation

of the decision undertaken by the classifier is required, the following idea could be used.

First, the value of the parameter p can be interpreted as the importance of the minority class relative to the majority class.

Second, if the value of the parameter s is close to 1, then RIONIDA classification can be interpreted in terms of rule-based method analogously as in the case of RIONA (see Subsection 3.3.5). As rules are preferable for human interpretation, one can also use the above interpretation in the cases when the value of the parameter s is not close to 1. We should only assume that switching value of the parameter s from the learned optimal value to 1 does not significantly change the quality of the algorithm (which can be detected during the learning phase). Then, an idea presented in Subsection 3.3.5 could also be used for generating rules using the parameter k.

If the value of the parameter s is close or less than 0 (and when switching the value of the parameter s to value 1 changes the quality in a significant way), then the behaviour of RIONIDA can be interpreted using the kNN method. Although in this case, it is difficult to interpret the parameters in the form of rules, the information that RIONIDA behaves as kNN together with the optimal value k can be quite informative for a human trying to understand the process of decision making.

4.6.2 Optimisation of the explicit performance measure

An important aspect of RIONIDA is that the performance measure it optimises is given explicitly. It is not a 'black box' regarding the optimised measure. In many algorithms for imbalanced data, the optimised measure is not given explicitly. Until experiments are performed, we do not always know in what aspects the given algorithm is satisfactory. For RIONIDA, we assume that there is given a performance measure, which we are going to optimise. In the current implementation of RIONIDA any measure defined over the confusion matrix can be easily used. This selected performance measure is optimised during the learning phase. In consequence, it is expected that the RIONIDA algorithm will perform with the high quality for unseen test examples concerning the pre-assumed performance measure.

This feature of RIONIDA can be very helpful in working with real-life applications. For example, let us assume that we have constructed a classifier for some domain and F-measure as the optimisation measure. However, after investigating results of the classifier, the user (e.g. the medical doctor) can reformulate the previous measure by adding some constraints like: *Sensitivity should be above the fixed threshold.* It is easy to redefine a measure with constraints of such types (defined over the confusion matrix) and then use RIONIDA to relearn the classifier with such new measure.

4.7 Conclusions for RIONIDA

Here, we summarise this chapter describing the newly developed algorithm RIONIDA. The remaining conclusions for RIONIDA, coming from the comparative experiments and extensive experimental analysis of this algorithm, are given at the end of the next chapter (see Section 5.6).

The RIONIDA algorithm is dedicated to imbalanced data. It is an extension of RIONA (and ONN). Thus, (i) it is based on a combination of instance-based learning and rule induction; (ii) the specific setting of RIONIDA parameters makes this algorithm equivalent to RIONA (or ONN); (iii) many conclusions for RIONA (see Section 3.8) are valid for RIONIDA, in particular, it does not require discretisation; it adequately groups values for both numerical and nominal attributes during rule generation. However, it uses two additional parameters (s and p) apart from the neighbourhood size (parameter k, analogous as for RIONA). RIONIDA uses more general rules than RIONA, namely scaled generalised local decision rules. These rules are parametrised with the parameter s. The value of s indicates the degree of rule-based approach (or inversely the degree of instance-based approach). The third parameter p is responsible for assigning relevant weights for the minority and majority classes.

RIONIDA uses (a fixed by a user) performance measure, relevant for imbalanced data, e.g. F-measure, or G-mean. For empirical justification of the components used in RIONIDA, we used the two mentioned above performance measures. We (empirically) showed (for these measures) that the neighbourhood size is a crucial factor for obtaining high value of performance measure (analogously as for RIONA). Additionally, we found that two other parameters (p, s) are also essential for obtaining high value of the performance measure by RIONIDA.

Thus, in the training phase, RIONIDA searches for the optimal triple values for all these three parameters. By the use of dynamic programming, the time complexity of this phase is relatively low. On the other hand, the space complexity can be noticeably high. Another critical aspect for fast performance of RIONIDA relates to limiting the size of sets of admissible values of three mentioned parameters. Also, we showed the possibility of reducing both the time and space complexity of RIONIDA training phase.

Also, for some specific settings of RIONIDA and some specific data sets (consisting of the borderline region only) we calculated the theoretical optimal values of the parameter p. The individual results for G-mean and F-measure are shown.

Additional important aspects of RIONIDA are that (i) the performance measure it optimises is given explicitly; (ii) the resulting classifier of RIONIDA can be interpreted in a way easily understandable by a human.

To sum up, RIONIDA is an extension of RIONA combining the instance- and rule-based approaches for imbalanced data. Additionally, RIONIDA combines these approaches in another aspect, namely by using more general than RIONA, special rules. All components of RIONIDA are essential for obtaining the high quality of its performance: optimisation of the fixed performance measure as well as three proposed internal parameters. Its performance is quick (both in the training and testing phase). Moreover, the theoretical results concerning the parameter responsible for assigning relevant weights for the minority and majority classes can be used for acceleration of the training phase.

Algorithm 13: getClassificationMatrixFast(trn, $trnSet$, K, P, S, $\{\varrho_a\}_{a\in A}$)

Input: currently considered example $trn \in trnSet$, training set $trnSet$, K, P, S – sets of admissible values for parameters k, p, s, respectively, family of pseudometrics for attributes $\{\varrho_a\}_{a\in A}$

Output: 3 dimensional matrix (for different parameters $(k,p,s) \in K \times P \times S$) of leave-one-out classification for trn

```
1  begin
2      k_max = |K| (we assume that K is the set of consequent natural numbers)
3      ϱ = Agr({ϱ_a}_{a∈A})
4      N = N(trn, trnSet \ {trn}, k_max, ϱ)
5      vector nn_1, ..., nn_{|N|} = N sorted according to the distance ϱ(trn, ·)
6      for k = 1 to |N| do
7          /* compute s_0 such that
                isConsistent(sg-rule (trn, nn_k, {ϱ_a}_{a∈A_sym}, s), N) is true for
                all s < s_0 and is false for all s ≥ s_0                    */
8          for i = 1 to |N| do
9              if d(nn_i) ≠ d(nn_k) then
10                 foreach a ∈ A do
11                     s_2(a) = the value s for which the value a(nn_i) is on the
                       border of the scaled interval (for numerical attributes) or
                       scaled ball (for symbolic attributes) of the sg-rule,
                       sg-rule (trn, nn_k, {ϱ_a}_{a∈A_sym}, s)
12                 end
13                 s_1(nn_i) = max_{a∈A} s_2(a) /* for s < s_1 object nn_i cannot cause
                       inconsistency of the sg-rule; for s ≥ s_1 the
                       sg-rule is inconsistent with object nn_i              */
14             end
15         end
16         s_0 = min_{1≤k≤|N|} s_1(nn_i) /* for s ≥ s_0 at least one training object
               breaks consistency of the rule                                */
17         for s ∈ S do
18             if s ≥ s_0 then nn_k.isConsistentOnLevel[s] = false;
19             else nn_k.isConsistentOnLevel[s] = true;
20         end
21     end
22     ...
23     return M
24 end
```

Algorithm 14: RIONIDA(*options*)

Input: *options* (including K, P, S and $optMeasure$) of the RIONIDA algorithm (we do not list all of them here; see Section 3.6 for more details)

Global variables:
A – conditional attributes ($A = A_{num} \cup A_{sym}$)
d – decision attribute
Agr – the aggregation of pseudometrics (appearing in Algorithms 9, 13; see Subsection 2.2.3)
Local variables:
$trnSet$ – training set
$\{\varrho_a\}_{a \in A}$ – family of pseudometrics for attributes
K, P, S – sets of admissible values for parameters k, p, s to be used during searching for k_{opt}, p_{opt}, s_{opt}, respectively
$optMeasure$ – optimisation measure from
{F-measure, G-mean, Accuracy}
. . . (local variables related to other options)
k_{opt}, p_{opt}, s_{opt} – the optimal values for k, p, s, respectively
begin
    $(K, P, S) = (options.K, options.P, options.S)$
    $optMeasure = options.optMeasure$
    by default Agr is defined according to Equation 2.1 (it may differ in case of choosing option for different weights for attributes – see Subsection 3.6.3)
    . . . (assignments related to other options)
end
Function *RIONIDA-train(trnSetDescription) : void*
    Input: *trnSetDescription* – description of training set together with the specifiaction of decision and conditional attributes
    using *trnSetDescription* specify the conditional attributes A, the decision attribute d and the training set $trnSet$
    foreach $a \in A_{num}$ **do**
        ϱ_a =normalised city-block metric based on $trnSet$ (see Equation 2.2)
    end
    foreach $a \in A_{sym}$ **do**
        ϱ_a = SVDM pseudometric based on $trnSet$ (see Equation 2.4)
    end
    . . . (operations related to other options)
    $(k_{opt}, p_{opt}, s_{opt})$ = findOptimalParams3D($trnSet$, K, P, S, $optMeasure$, $\{\varrho_a\}_{a \in A}$) (see Algorithm 12)
end
Function *RIONIDA-classify(tst) : decision*
    Output: predicted decision for tst
    return RIONIDA-classify(tst, $trnSet$, k_{opt}, p_{opt}, s_{opt}, $\{\varrho_a\}_{a \in A}$) (see Algorithm 9)
end

Chapter 5

Experiments and results

This chapter discusses the results of performed experiments using the RIONIDA algorithm, which was described in Chapter 4. The aim is to analyse the algorithm performance and compare it with the performance of its predecessor, i.e. RIONA, and also with some of the state-of-the-art algorithms designed for imbalanced data available (together with their codes) for the author of the thesis. The whole experimental environment, including the code, data sets and short launching instruction, which allows to easily reproduce the most important experiments, is available for the use of reviewers on request[1].

The chapter is divided into five sections. Section 5.1 describes the general experimental setup. Section 5.2 presents learning algorithms and filters used in the comparative experiments as well as different variants of the experimental settings. The most important part of this chapter, Section 5.3, discusses the performance of the RIONIDA algorithm relative to some selected algorithms known from the literature. One could move the next two complementary sections to Appendix of this thesis. However, we keep them here because they contain a continuation of considerations of Section 5.3. Section 5.4 presents some additional comments helping to understand the results from the previous section and advantages of RIONIDA. Finally, Section 5.5 presents some additional experimental results, which can help the readers to understand more deeply why RIONIDA outperforms RIONA (with filter) and BRACID. Moreover, this section presents some additional experiments for verifying if RIONIDA could be even more improved.

5.1 General experimental setup

This section briefly revisits, taking into account the discussion made in Section 2.6, and presents more precisely how the experiments, described in this thesis, were designed. The following Subsections 5.1.1-5.1.5 are related to Subsections 2.6.1-2.6.5, respectively. The specific group of algorithms which were compared with RIONIDA is discussed in Section 5.2.

[1] ggora@mimuw.edu.pl

5.1.1 Performance measure

To evaluate learning algorithms (and as sub-task also classifiers), we use two performance measures, namely F-measure and G-mean. We have not used AUC measure. One reason is the criticism about it (for the references see Subsection 2.6.1). The second is that BRACID, one of the important learning algorithms that we wanted to compare with, does not return probabilities for the two decision classes (only the deterministic decision is returned)[2].

5.1.2 Estimation of the chosen performance measure

For estimation of the chosen performance measure (out of two mentioned above), we use 10 times repeated 10-fold stratified cross-validation. Partial results of each 10-fold stratified cross-validation are micro-averaged. As the final estimation of the desired measure, the average of ten repetitions of this procedure is used. In Figure 5.2, we summarise how the estimation of the chosen performance measure is computed. This figure uses the notion of *AF-learner*, which will only be defined in Subsection 5.2.1. However, at this stage, the readers can think of AF-learner as simply a learning algorithm.

For all compared learning algorithms, the same splits in the cross-validation process are used. It can be thought that the estimation is done in parallel for all learning algorithms. In practice, we simply use the same random seed (used for random partitions of data sets) for all learning algorithms in the process of estimating the chosen measure.

5.1.3 Selection of data sets for evaluation

In order to perform comparative experiments, several data sets have been selected. All but one are from the UCI Machine Learning Repository [131]. Only *mammography* data set is not publicly available and was supported by Nitesh Chawla [37] (see also [224]).

Data description

In this section, data sets used in experiments are presented. Data sets for the binary classification task are relevant directly for the RIONIDA algorithm. In turn, data sets containing originally more than two classes were transformed into the binary classification task by choosing one small class or joining several small classes into one (minority) class; other classes were joined into another (majority) class. Table 5.1 presents all data sets used in the experiments.

We give here a very brief description of the used data sets according to the information gained from the UCI repository and also about *mammography* data set. We divide it into two parts. In the first part, we shortly describe data sets with

[2] In fact, in the current implementation, RIONIDA also does not provide the possibility to return probabilities for the two decision classes. However, this can be easily implemented if it is only needed.

binary decisions which can be directly used for the binary classification task. Below, we list these data sets together with their short description.

1. Breast Cancer Data Set – This is one of the three data sets provided by the Oncology Institute that has repeatedly appeared in the ML literature. This data set includes instances of two (decision) classes: *no recurrence events* and *recurrence events*.

2. Breast Cancer Wisconsin (Original) Data Set – The classification task is to separate *benign* samples from *malignant* ones on the basis of nine diagnostically important cytological characteristics.

3. Statlog (German Credit Data) Data Set – This data set classifies people described by a set of attributes as *good* or *bad* credit risks. It originally also comes with a cost matrix: it is worse to classify a customer as good when they are bad (cost 5) than to classify a customer as bad when they are good (cost 1).

4. Haberman's Survival Data Set – The data set contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer. The classes to predict are: *the patient survived five years or longer* and *the patient died within five years*.

5. Hepatitis Data Set – The data was provided by Dr Peter Gregory of Stanford University's School of Medicine. The scientific problem involves 155 acute chronic hepatitis patients. Of these, 33 were observed to *die* from the disease while 122 *survived*. Dr Gregory aimed to understand the effect of the measured variables like age, sex, and standard chemical measurements on the chance of patient survival.

6. Ionosphere Data Set – This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. *Good* radar returns are those showing evidence of some type of structure in the ionosphere. *Bad* returns are those that do not; their signals pass through the ionosphere. Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this data set are described by two attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal.

7. Microcalcifications in Mammography – Mammography images are transformed into 6 numerical attributes. We have two decisions: *normal pixels in image* (the majority class) and *abnormal pixels in image* (the minority class).

8. Pima Indians Diabetes Data Set – The diagnostic, binary-valued variable investigated is whether the patient shows signs of diabetes according to World Health Organisation criteria (i.e. if the 2-hour post-load plasma glucose was at

least 200 mg/dl at any survey examination or if found during routine medical care).

9. Blood Transfusion Service Center Data Set – The center passes their blood transfusion service bus to one university in Hsin-Chu City in Taiwan to gather blood donated about every three months. 748 donors were selected at random from the donor database. These 748 donor data, each one included R (Recency – months since the last donation), F (Frequency – total number of donation), M (Monetary – total blood donated), T (Time – months since the first donation), and a binary variable representing whether he/she *donated blood in March 2007* (or *not donated*).

The second part contains data sets for the multiple-class classification task. These data sets cannot be directly used for binary classification tasks. In preprocessing, they were transformed into data sets with binary decisions[3]. Usually, it is done by choosing one small class as the minority class, and other classes are joined and treated as the majority class. Below, we list these data sets together with their short descriptions.

1. Abalone Data Set – Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem. *Rings* is the decision attribute (+1.5 gives the age in years) and contains integer values between 1 and 29. In the binary classification task, classes 1-4 and 16-29 were joined into one class (the minority class).

2. Balance Scale Data Set – This data set was generated to model psychological experimental results. Each example is classified as having the balance scale *tip to the right*, *tip to the left*, or be *balanced*. The attributes are the left weight, the left distance, the right weight, and the right distance. The correct way to find the class is the greater of (left-distance · left-weight) and (right-distance · right-weight). If they are equal, it is balanced. In the binary classification task, a class labelled by *balanced* was chosen as the minority class.

3. Car Evaluation Data Set – Car Evaluation Database was derived from a simple hierarchical decision model originally developed for the demonstration of DEX, an expert system for decision making. The model evaluates cars according to the given concept structure (e.g. higher-level attribute comfort depends on low-level attributes: doors, persons, lug boot). This data set contains examples with the structural information removed, i.e. directly relates car to the six input attributes: buying, maint, doors, persons, lug boot, safety. The decision attribute contains values: *unacc*, *acc*, *good*, *v-good*. In the binary classification task, a class labelled by *good* was chosen as the minority class.

[3]It should be borne in mind that the classification of multiple-class imbalanced data is a separate problem with which we do not deal in the thesis (see Introduction).

4. Heart Disease Data Set – This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland data set is the only one that has been used by ML researchers. The decision attribute refers to the presence of heart disease in the patient. It is integer-valued from 0 (no presence) to 4. Experiments with the Cleveland data set usually had focus on to distinguish presence (values *1,2,3,4*) from absence (value *0*). In the binary classification task, we try to distinguish heart disease indicated by number *3* from other states.

5. Ecoli Data Set – The aim is to predict protein localisation sites in gram-negative bacteria, given the amino acid sequence information alone. There are seven localisation sites (decisions): cytoplasm (*cp*), 4 kinds of the inner (cytoplasmic) membrane (*im*, *imU*, *imL*, *imS*), the periplasm (*pp*), and 2 kinds of the outer membrane (*om*, *omL*). In the binary classification task, the aim is to distinct localisation site *imU* (inner membrane, uncleavable signal sequence) from others.

6. Glass Identification Data Set – The original task is to determine whether the glass was a type of 'float' glass or not. The study of the classification of types of glass was motivated by the criminological investigation. At the scene of the crime, the glass left can be used as evidence if it is correctly identified. Decision class may contain seven values: *building windows float processed*, *building windows non-float processed*, *vehicle windows float processed*, *vehicle windows non-float processed* (none in this data set) and 3 other decisions with numbers *5*, *6*, *7* indicating for the rest types. In the binary classification task, the aim is to discern *vehicle windows float processed* type of glass from other types.

7. Thyroid Disease Data Set – One of Thyroid databases is used, i.e. database donated by Stefan Aberhard (Thyroid gland data). Five laboratory tests are used to try to predict whether a patient's thyroid belongs to the class *euthyroidism*, *hypothyroidism* or *hyperthyroidism*. The diagnosis (the class label) was based on a complete medical record, including among others anamnesis and scan. In the binary classification task, the aim is to discern *hyperthyroidism* from other classes.

8. Nursery Data Set – This data set was derived from a hierarchical decision model originally developed to rank applications for nursery schools. The final decision depended on three subproblems: the occupation of parents and child's nursery, family structure and financial standing, and social and health picture of the family. The hierarchical model ranks nursery-school applications according to the given concept structure. The decision belongs to the following ones: *not_recom*, *recommend*, *very_recom*, *priority*, *spec_prior*. The Nursery Data Set contains examples with the structural information removed, i.e. directly relates the decision to the eight input attributes. In the binary classification task, the aim is to discern *very_recom* from other classes.

9. Post-Operative Patient Data Set – The classification task of this data set

is to determine the decision related to patients in a postoperative recovery area: where they should be sent next. Because hypothermia is a significant concern after surgery, the attributes correspond roughly to body temperature measurements. Possible decisions are as follows: *patient sent to the Intensive Care Unit*, *patient prepared to go home*, and *patient sent to the general hospital floor*. In the binary classification task, the aim is to discern class *patient prepared to go home* from other classes.

10. Statlog (Vehicle Silhouettes) Data Set – The purpose is to classify a given silhouette as one of four types of vehicle (*opel*, *saab*, *bus*, *van*), using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles. In the binary classification task, the aim is to discern *van* from other classes.

11. Yeast Data Set – The aim is to classify proteins into their various cellular localisation sites based on their amino acid sequences. The classes are the following: *CYT* (cytosolic or cytoskeletal), *NUC* (nuclear), *MIT* (mitochondrial), *ME3* (membrane protein, no N-terminal signal), *ME2* (membrane protein, uncleaved signal), *ME1* (membrane protein, cleaved signal), *EXC* (extracellular), *VAC* (vacuolar), *POX* (peroxisomal), *ERL* (endoplasmic reticulum lumen). In the binary classification task, the aim is to discern *ME2* (membrane protein, uncleaved signal) from other classes.

Among these data sets used in the experiments, a few are inconsistent (e.g. *breast-cancer*, *haberman*, *mammography*, *postoperative*, *transfusion*). Among these data sets, a few contain missing values (e.g. *breast-cancer*, *breast-w*, *hepatitis*, *cleveland*, *postoperative*).

Table 5.1: Description of data sets used in experiments.

Data set name	Identifier	No of examples	No of conditional attributes (numerical, nominal)	No of original classes	Classes for binary classification task (minority class, majority class)	Minority class (in %)
Abalone	abalone	4177	8 (7, 1)	29	(1-4 and 16-29, others)	8.02
Balance Scale	balance-scale	625	4 (0, 4)	3	(B=balanced, others)	7.84
Breast Cancer	breast-cancer	286	9 (0, 9)	2	(recurrence-events, no-recurrence-events)	29.72
Breast Cancer Wisconsin (Original)	breast-w	699	9 (9, 0)	2	(malignant, benign)	34.48
Car Evaluation	car	1728	6 (0, 6)	4	(good, others)	3.99
Heart Disease (Cleveland)	cleveland	303	13 (6, 7)	5	(3, others)	11.55
Statlog (German Credit Data)	credit-g	1000	20 (7, 13)	2	(bad, good)	30.00
Ecoli	ecoli	336	7 (7, 0)	8	(imU, others)	10.42
Glass Identification	glass	214	9 (9, 0)	7	(3=vehicle_windows_f_p, others)	7.94
Haberman's Survival	haberman	306	3 (3, 0)	2	(1=the patient survived, 2=died)	26.47
Hepatitis	hepatitis	155	19 (6, 13)	2	(1=die, 2=live)	20.65
Ionosphere	ionosphere	351	34 (34, 0)	2	(bad, good)	35.90
Microcalcifications in Mammography	mammography	11183	6 (6, 0)	2	(1=abnormal pixels, 0=normal pixels)	2.33
Thyroid Disease	new-thyroid	215	5 (5, 0)	3	(2=hyper, others)	16.28
Nursery	nursery	12960	8 (0, 8)	5	(very_recom, others)	2.53
Pima Indians Diabetes	pima	768	8 (8, 0)	2	(1=tested positive for diabetes, 0)	34.90
Post-Operative Patient	postoperative	90	8 (0, 8)	3	(S=patient prepared to go home, others)	26.67
Blood Transfusion Service Center	transfusion	748	4 (4, 0)	2	(1=donated blood, 0=not donated)	23.80
Statlog (Vehicle Silhouettes)	vehicle	846	18 (18, 0)	4	(van, others)	23.52
Yeast	yeast	1484	8 (8, 0)	10	(ME2, others)	3.44

Argumentation for the choice of benchmarks

This choice of data sets seems to create a relevant base for experiments since we have selected 20 fairly diverse imbalanced data sets considering the aspects described below.

- The size of data sets is varied (from 90 to 11180 examples in total).
- The percentage of the minority class is varied (from 2.33% to 35.90%, i.e. imbalance ratio is between around 2 and 42).
- The types of attributes are also varied (either only numerical, either only symbolic or mixed numerical and symbolic).
- The data set difficulty, in terms of types of examples discussed in Subsection 2.4.3, is also varied. This fact is discussed in [154] where most of the data sets which we use in our experiments were inspected. For example, out of the data sets inspected in [154] and occurring in our experiments the most difficult data sets are: (sorted in order from the 'most difficult' to 'easier ones') *balance-scale*, *yeast*, *transfusion*, *postoperative*, *abalone*, *glass*, *cleveland*. These data sets contain a small number (or even none) of safe examples, more than 25% of outlier examples and a relatively high number of the border or rare examples. For example, *balance-scale* data set contains no safe example, and no borderline example, 8.16% of rare examples and 91.84% of outlier examples; *cleveland* data set contains no safe example, 31.43% borderline examples, 17.14% of rare examples, and 51.43% of outlier examples (for details and information about other data sets see [154]).
- There are both consistent and inconsistent data sets. There are data sets with and without missing values.

5.1.4 Statistical tests

We use the Friedman statistical test (see Subsection 2.6.4) for comparing multiple learning algorithms on multiple data sets. If this test passes, then we use the post-hoc tests Nemenyi or Finner (for a discussion about both of them see Subsection 2.6.4). The first one enables us to compare all learning algorithms against each other, and the second one is used to compare the RIONIDA learning algorithm with other algorithms used in the comparative experiments. For all tests, we use the significance level $\alpha = 5\%$.

The statistical analysis was done using the R Project for Statistical Computing, commonly known as the R Package (see [170]).

5.1.5 Selecting the best learning algorithm for real-life data sets

Additionally to data sets from the UCI repository we also chose the *mammography* data set (not included in the UCI repository). This gives us a greater variety of

imbalanced data sets and gives us more persuasive arguments for our conclusions. However, taking into account the remarks in Subsection 2.6.5, one should be aware that our comparison can give us only some suggestions about the quality of the new proposed algorithm (RIONIDA).

5.2 Learning algorithms and filters used in comparative experiments

Generally, one of the aims of performed experiments was to compare the new algorithm (RIONIDA) with some other state-of-the-art learning algorithms. Some of them are specially designed for the classification of imbalanced data, and some are not. However, one can also use state-of-the-art learning algorithms developed for balanced data and apply them to the results of sampling methods (filters) dedicated to imbalanced data. We use two types of well-known filters (and additionally one trivial `Null-filter` for cases when no filter is used). Below we describe all learning algorithms and filters used in the experiments.

Moreover, we describe the variants of these algorithms and filters used in the experiments. Taking into account the variety of learning algorithms use, there arise different possibilities of the comparison of algorithms. Thus, we also present three strategies for selecting representative *scores* for learning algorithms used in the experiments. These strategies (used later for comparisons of learning algorithms) are related to three levels of increasing challenge for RIONIDA in relation to the other algorithms.

5.2.1 Configuration and AF-learner

One could perform comparative experiments using the default options for learning algorithms and one selected specific filter, e.g. well-known filter `SMOTE`. Those interested in such standard comparison only can skip most of the below considerations and move on to respective fragments of Subsections 5.3.1 and 5.3.2 related to default settings (starting on page 164 and 175, respectively). However, it could be not satisfactory as one could argue that for different use of learning algorithms (different options settings) or different use of filters the performance of algorithms could change and as a result could change final conclusions. Thus, we decided to make a comparative study taking into account many possible combinations of options for used learning algorithms. Moreover, any learning algorithm with the specific combination of options can be preceded by data preprocessing with the use of different filters. In our experiments, we use for each learning algorithm a significant number of combinations of options combined with a few different filters. However, we try to design experiments to make a general comparison of the selected learning algorithms (taking into account their different settings and use of different filters).

For the sake of readability, we use the following nomenclature for the description of performed experiments. *Options* are some specific parameters of learning algorithms to be specified *a priori* by the user, which may change the behaviour of algorithms. The fixed specific arrangement of options (with specific values, if needed) for the

learning algorithm is called the *configuration of the algorithm*. The fixed specific use of filters (used for data preprocessing before running an algorithm) is called the *configuration of filters*. The set of configurations of the algorithms and the set of configurations of filters used in the experiments are presented in Subsections 5.2.3 and 5.2.4.

The learning algorithm with the fixed configuration of the algorithm and the fixed configuration of filters used before running the algorithm is called *AF-learner*. Figure 5.1 illustrates the idea of AF-learner. The input to AF-learner is a training set, and the output is a classifier. One can think of AF-learner as an extended learning algorithm with specified its own options and specified additional options determining which filter to use in the preprocessing phase of the algorithm. Thus, AF-learner is defined by a pair $\langle algConf, filterConf \rangle$, where $algConf$ is an algorithm name together with a configuration of the algorithm, and $filterConf$ is a configuration of filters. In the following considerations, sometimes we identify such a pair with its corresponding AF-learner.

It should be noted that any learning algorithm selected for experiments, formally defines a class of learning algorithms (taking into account the setting of its options and the possible use of preprocessing filters). The notion of AF-learner was introduced to reduce the ambiguity of terms and make our considerations more precise. Also, any learning algorithm selected for comparisons (with a purpose to establish its specific AF-learners) from now on will often be called the algorithm, for short.

For each algorithm used in the experiments, several AF-learners are available. The main idea of the following considerations is to compare the RIONIDA algorithm not only with one *a priori* chosen AF-learner for each of algorithms used in the experiments, but with many of them, and what is more, with the 'optimal combination' of AF-learners defined by the specific algorithm. In other words, we wanted to make the 'best' use of the algorithms and filters selected for comparisons with RIONIDA. One could make comparisons with all chosen AF-learners. However, it would be inconvenient for presentation (we have chosen 99 AF-learners as it will be explained later), not to mention other difficulties. Moreover, such an approach would not enable us to use in comparisons the mentioned 'optimal combination' of these AF-learners.

We have decided to use for the final comparison of each algorithm the scores (the estimation of the chosen performance measure; see Subsection 5.1.2) of its relevant AF-learners. In the next subsections, we specify algorithms and their AF-learners used for comparisons. Also, we specify three strategies for selecting the representative scores for the algorithms (related to three levels giving other algorithms an increasing advantage over RIONIDA).

5.2.2 Algorithms used in comparative experiments

In our comparative experiments, we used 3 algorithms developed especially for imbalanced data and 7 algorithms developed for balanced data. The performance of the latter can be significantly improved for imbalanced data by using them together with filters developed for such data sets.

Below, we list three algorithms used in the experiments which were specially

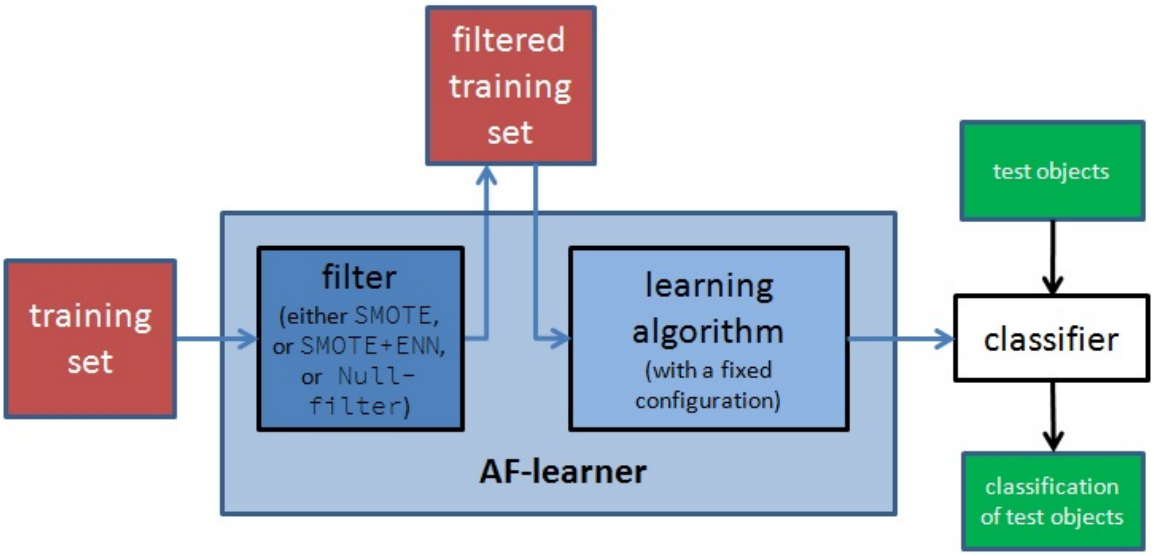


Figure 5.1: Illustration of the idea of AF-learner.

developed for the analysis of imbalanced data together with their short descriptions.

1. BRACID (Bottom-up induction of Rules And Cases for Imbalanced Data) – Analogously to RISE (see the description of RISE below) it uses an integrated representation of rules and single instances. It comprehensively addresses the issues associated with imbalanced data. It uses the strategy of bottom-up induction of rules from single examples with the specific generalisation strategy. A conflict resolution is based on the supports of the nearest rules to the test example. For more information, see [152, 153].

2. MODLEM-C – an extension of MODLEM algorithm (see below) with the possibility to strengthen Sensitivity. The rule strength is multiplied for all rules describing the minority class by the same real number called the *strength multiplier* given as a parameter. It is equivalent to adding duplicates from the minority class from the training set. For more information, see [89, 90] (and citations given for MODLEM below).

3. RIONIDA (Rule Induction with Optimal Neighbourhood for Imbalanced Data Algorithm) – algorithm described in the thesis.

Below, we list the remaining algorithms (generally dedicated to balanced data) used in the experiments with their short descriptions.

1. kNN[4] (k-nearest neighbours learning algorithm) – Can select the relevant value of k based on cross-validation. Can also do distance weighting. For more information, see [7].

[4]It should be mentioned that the kNN algorithm used in the comparative experiments comes from WEKA library. It may differ in details (e.g. see Definition 2.14 of the neighbourhood) in comparison to the kNN algorithm shown in Subsection 2.3.3 (see Algorithm 3). However, we do not want to go into details of these differences.

2. MODLEM – Heuristic algorithm generating a minimal set of rules. It is a kind of extension of algorithm LEM2 [86] to work with numerical attributes with no need of discretisation in preprocessing. For numerical attributes, it uses similar elementary conditions as in decision trees (value of attribute less or greater than a given value). For more information, see [188–190].

3. J48 (decision tree learning algorithm) – Class for generating a pruned or unpruned C4.5 decision tree. For more information, see [169] (see also [216]).

4. PART – The method combines the two rule learning paradigms: used by C4.5 and RIPPER. It is called PART because it is based on partial decision trees. PART uses separate-and-conquer, builds a partial C4.5 decision tree in each iteration and makes the 'best' leaf into a rule. For more information, see [68] (see also [216]).

5. RIPPER (Repeated Incremental Pruning to Produce Error Reduction) – This algorithm implements a propositional rule learner proposed by William W. Cohen as an optimised version of IREP (see [71]). For more information, see [42] and also [216].

6. RISE (Rule Induction from a Set of Exemplars) – This algorithm is a unification of the two widely-used empirical approaches: rule induction and instance-based learning. In this algorithm, instances are treated as maximally specific rules, and classification is performed using the best match strategy. Rules are learned by gradually generalising instances until no improvement in apparent Accuracy is obtained. For more information, see [53].

7. RIONA (Rule Induction Optimal Neighbourhood Algorithm) – algorithm described in the thesis.

Table 5.2 provides technical details about these algorithms.

5.2.3 Configurations of algorithms used in comparative experiments

Let us recall that for a given algorithm its configuration is defined by the specific combination of options with their specific values. Generally, for any algorithm, the set of all possible configurations of the algorithm can be potentially huge. For any algorithm, we would like to consider these sets that are reasonably reduced and simultaneously representing relevant variations of the algorithm performance. Thus, for each algorithm, we selected a specific set of configurations of the algorithm which were used in the experiments. It was done in two steps.

First, for each algorithm, we selected the set of options for which varied settings were used. This choice was done *a priori*. However, we considered options which – used with non-default settings – could potentially improve the algorithm performance in the considered classification problem. Table 5.3 presents the selected options of algorithms with their descriptions. Options not listed in this table are used with their default values for the following comparative experiments.

Table 5.2: References concerning algorithms used in comparative experiments. Each entry in the last column indicates whether the algorithm is developed for imbalanced data (ID).

algorithm short name	author(s) of idea	author(s) of used implementation	additional information	for ID
BRACID	Jerzy Stefanowski, Krystyna Napierała [153]	Krystyna Napierała	not publicly available	yes
MODLEM	Jerzy Stefanowski [188]	Szymon Wojciechowski	available in official WEKA package	no
MODLEM-C	Jerzy Stefanowski, Jerzy Grzymala-Busse [90]	Szymon Wilk	not publicly available	yes
RIONA	Arkadiusz Wojna, Grzegorz Góra [82]	Arkadiusz Wojna, Grzegorz Góra	available in library Rseslib [1] and in official WEKA package	no
RIONIDA	Grzegorz Góra	Grzegorz Góra	not yet publicly available (planned to be publicly available as RIONA)	yes
RISE	Pedro Domingos [53]	Krystyna Napierała	reimplementation of the original author's implementation	no
kNN	David W. Aha, Dennis Kibler, Marc K. Albert [7]	Stuart Inglis, Len Trigg, Eibe Frank	from WEKA library: weka.classifiers.lazy.Ibk	no
J48	Ross Quinlan [169]	Eibe Frank	from WEKA library: weka.classifiers.trees.J48	no
PART	Eibe Frank, Ian H. Witten [68]	Eibe Frank	from WEKA library: weka.classifiers.rules.PART	no
RIPPER	William W. Cohen [42]	Xin Xu, Eibe Frank	from WEKA library: weka.classifiers.rules.Jrip	no

Second, we selected several combinations of the (selected) options. In particular, these combinations are defined by a selected set of values to be used for options with their parameter values. In this way, the configurations of the algorithms were selected. This choice was also done *a priori*. However, for any algorithm, we considered a 'reasonable' set of its configurations. Let us also note that for any algorithm, its default combination of options was included in the set of its configurations (if only such a default combination was specified for the algorithm). Table 5.4 presents for each algorithm its selected configurations used in the experiments.

In the sequel, we write RIONIDA$_{\mathrm{G}}$ instead of RIONIDA `-T 0`, and RIONIDA$_{\mathrm{F}}$ instead of RIONIDA `-T 1` (i.e. the performance measure to be optimised in RIONIDA is set to G-mean or F-measure, respectively) as it was told in Table 5.3. Let us note that for the RIONIDA algorithm, we use only one configuration while making comparisons for G-mean and one configuration while making comparisons for F-measure (depending on the considered performance measure). It means that in the experiments for the fixed performance measure, we simply use either RIONIDA$_{\mathrm{G}}$ or RIONIDA$_{\mathrm{F}}$. While describing the experiments, RIONIDA denotes RIONIDA$_{\mathrm{G}}$ or RIONIDA$_{\mathrm{F}}$ depending on the chosen performance measure (G-mean or F-measure, respectively).

Also, we have chosen only one configuration for the RIONA algorithm. RIONA has, of course, more possible configurations which could give a better result. However, we used only those options which are analogous to the options of the RIONIDA algorithm. The main aim of using RIONA in the comparative experiments was to check whether RIONA with the relevant filters only can be competitive with RIONIDA. A more detailed performance comparison of algorithms RIONIDA and RIONA is presented in Subsection 5.5.2.

5.2.4 Configuration of filters used in comparative experiments

In our experiments, two filters were used:

- `SMOTE` (Synthetic Minority Over-sampling Technique) – see description in Subsection 2.5.1. For more information, see [37];
- `ENN` (Edited Nearest Neighbour) – tries to discard unreliable majority examples, by removing any majority examples whose class label differs from the class of at least two of its three nearest neighbours. For more information, see [214] (see also [14]).

Table 5.5 provides technical details about these filters.

Any fixed specific combination of the above filters is called the *configuration of filters*. In the experiments, we use the following configurations of filters:

1. `Null-filter` (no filter) – in such case no preprocessing of data is performed (original training data are used).
2. `SMOTE` – training data are preprocessed by filter `SMOTE` so that new training data set is presented to the algorithm.
3. `SMOTE+ENN` – first filter `SMOTE` is used, and then filter `ENN` is used.

Table 5.3: Technical details concerning options of algorithms used in experiments.

algorithm	used options	description
kNN	`-K` [k]	Number of nearest neighbours (k) used in classification (default 1) or specific use in case the option `-X` is used.
	`-X`	Used for selecting the optimal number of nearest neighbours between 1 and the k value (in this case, this value is specified by option `-K`) using leave-one-out evaluation on the training data (used when k > 1).
PART	`-U`	Generate unpruned decision list.
	`-R`	Use reduced error pruning.
	`-C` [pruning confidence]	Set the confidence threshold for pruning (default 0.25).
J48	`-U`	Use the unpruned tree.
	`-A`	The Laplace smoothing for predicted probabilities.
	`-C` [pruning confidence]	Set the confidence threshold for pruning (default 0.25).
RIPPER	`-P`	Whether NOT use pruning (default: use pruning).
	`-E`	Whether NOT check the error rate ≥ 0.5 in stopping criteria (default: check).
RISE		(no options are used)
MODLEM		(no options are used)
MODLEM-C	`-M` [strength of min class]	Used for setting the constant strength multiplier for the minority class.
RIONA		(no options are used – all options settings are the same as default options in the RIONIDA algorithm)
BRACID		(no options are used)
RIONIDA	`-T` [optimisation measure]	Set performance measure to be optimised (0=G-mean; 1=F-measure; 2=Accuracy). It should be stressed that this option is not used in the experiments to select the optimal AF-learner. It is fixed and set depending on the performance measure in which we are interested in the given experiment. For clarity and for short we will write $\text{RIONIDA}_{\text{G}}$ instead of RIONIDA `-T 0`, and $\text{RIONIDA}_{\text{F}}$ instead of RIONIDA `-T 1`. (no other options are used – for other options default settings are used)

Table 5.4: Technical details of all selected configurations for each algorithm.

algorithm	**used config.**	**configuration description (the meaning of the combination of options)**

kNN	`-K 1`
	`-K 2`
	...
	`-K 10`
		Use in classification constant number of nearest neighbours equal to 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10.

	`-K 100 -X`	Select the number of nearest neighbours between 1 and 100 (fixed in the specification) using leave-one-out evaluation on the training data.
PART	`-U`	Unpruned decision list.
	`-R`	Use reduced error pruning.

	`-C 0.5`
	`-C 0.25`
	`-C 0.1`
		Set confidence threshold for pruning to 0.5, 0.25 (default) or 0.1.

J48	`-U`	Use the unpruned tree.
	`-A`	Laplace smoothing for predicted probabilities.
	`-A -U`	Both above settings.

	`-C 0.5`
	`-C 0.25`
	`-C 0.1`
		Set confidence threshold for pruning to 0.5, 0.25 (default) or 0.1.

RIPPER		Default options.
	`-P`	Do not use pruning.
	`-E`	Do not check the error rate ≥ 0.5 in stopping criteria.
	`-E -P`	Both above settings.
RISE		Default options.
MODLEM		Default options.

MODLEM-C	`-M 1`
	`-M 2`
	...
	`-M 10`
		Constant strength multiplier for the minority class equal to 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10.

RIONA		options set to the default options of the RIONIDA algorithm
BRACID		Default options.
RIONIDA	(`-T 0` or `-T 1`)	Performance measure to be optimised is fixed (G-mean or F-measure, i.e. specified by option `T`) in the current experiment. For other options, their default values are used (CSVDM distance measure, none method for attribute weighting, use of indexing to accelerate nearest neighbours search, find the optimal number of nearest neighbours, the maximum number of neighbours while optimising automatically equal to 100, use rules to filter nearest neighbours, votes to neighbours does not depend on distance)

`SMOTE` is one of the most popular over-sampling methods for imbalanced data with quite good performance in comparison to other sampling methods. Thus, it is frequently used as a counterpart in empirical evaluations (see e.g. [14]).

The motivation for selecting the configuration of filters `SMOTE+ENN` comes from [14], where it was shown that this combination of filters provides in practice very good performance in comparison to other combination of filters for data sets with a small number of positive examples. Such a combination of filters is often applied in the literature in case of the data sets with complex distributions of classes (see e.g. [152]).

Table 5.5: References concerning filters used in comparative experiments.

filter short name	author(s) of idea	author of used implementation	used implementation
`Null-filter`			weka.filters.AllFilter
`SMOTE`	Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer [37]	Tomasz Maciejewski [141]	implemented in the Stefanowski group
`ENN`	Dennis L. Wilson [214]	Michał Marcinkowski [145]	implemented in the Stefanowski group

5.2.5 AF-learners used in comparative experiments

Let us recall that any AF-learner is defined by a pair consisting of a configuration of the algorithm (together with the algorithm name) and a configuration of filters. As it was mentioned, we use the fixed set of selected configurations of the algorithm (see Subsection 5.2.3) and the fixed set of selected configurations of filters (see Subsection 5.2.4).

For a fixed algorithm, the set of possible AF-learners corresponds to the Cartesian product of two sets: the set of selected configurations of the algorithm and the set of selected configurations of filters. However, for the experiments, we take a subset of this set depending on whether the algorithm is dedicated to imbalanced data or not. Certainly, one can assume that filters cannot significantly improve the quality of algorithms dedicated to imbalanced data. Therefore, for algorithms not dedicated to imbalanced data, all 3 possible configurations of filters are used, and for those dedicated to imbalanced data no filter is used (i.e. `Null-filter` is used as the configuration of filters).

Table 5.6 summarises the information about AF-learners used in the experiments for each algorithm. Although the information in this table is redundant, it is presented

for clarity. The existence of value *no* in the second column (i.e. algorithm is not dedicated to imbalanced data) is equivalent to the existence of value 3 in the fourth column (i.e. 3 configurations of filters are used), and the existence of value *yes* in the second column (i.e. algorithm is dedicated to imbalanced data) is equivalent to the existence of value 1 in the fourth column (i.e. one configuration of filters is used, namely `Null-filter`). The number of AF-learners is equal to the product of the number of configurations of the algorithm and the number of configurations of filters.

Let us consider an example for all AF-learners used in the experiments for one exemplary algorithm. First, let us look for the selected configurations of the PART algorithm in Table 5.4 (see also Table 5.6). The 5 selected configurations of this algorithm are as follows: PART `-U`, PART `-R`, PART `-C 0.5`, PART `-C 0.25`, or PART `-C 0.1`. Since this algorithm is not dedicated to imbalanced data, we use 3 configurations of filters, namely `Null-filter`, `SMOTE` and `SMOTE+ENN`. We obtain for this algorithm the following set of AF-learners to be used in the experiments:
⟨PART `-U`, `Null-filter`⟩, ⟨PART `-U`, `SMOTE`⟩, ⟨PART `-U`, `SMOTE+ENN`⟩,
⟨PART `-R`, `Null-filter`⟩, ⟨PART `-R`, `SMOTE`⟩, ⟨PART `-R`, `SMOTE+ENN`⟩,
⟨PART `-C 0.5`, `Null-filter`⟩, ⟨PART `-C 0.5`, `SMOTE`⟩, ⟨PART `-C 0.5`, `SMOTE+ENN`⟩,
⟨PART `-C 0.25`, `Null-filter`⟩, ⟨PART `-C 0.25`, `SMOTE`⟩,
⟨PART `-C 0.25`, `SMOTE+ENN`⟩,
⟨PART `-C 0.1`, `Null-filter`⟩, ⟨PART `-C 0.1`, `SMOTE`⟩, ⟨PART `-C 0.1`, `SMOTE+ENN`⟩.

Let us note that for both the algorithms RIONIDA and BRACID, the set of AF-learners consists of one element (i.e. these algorithms are used with their default options and without filter).

Table 5.6: For each algorithm, there are given: (i) information whether the algorithm is dedicated to imbalanced data, (ii) number of configurations of the algorithm, (iii) number of configurations of filters, and (iv) number of AF-learners used in experiments.

algorithm	**is dedicated to imbalanced data?**	**number of configurations of the algorithm**	**number of configurations of filters**	**number of AF-learners**
kNN	no	11	3	33
PART	no	5	3	15
J48	no	6	3	18
RIPPER	no	4	3	12
RISE	no	1	3	3
MODLEM	no	1	3	3
MODLEM-C	yes	10	1	10
RIONA	no	1	3	3
BRACID	yes	1	1	1
RIONIDA	yes	1	1	1

5.2.6 Selection of the representative scores for learning algorithms

This subsection is crucial for understanding what exactly will be presented as the performance results of the algorithms used in the comparative experiments (in Section 5.3). As it was mentioned, any algorithm used in experiments, formally, defines a class of learning algorithms (defined by settings of options of the algorithm and preprocessing filters). We call these specific learning algorithms AF-learners. Let us recall that for each algorithm we selected a reasonable (in size and representability) subclass of such class, i.e. the set of AF-learners used in the experiments (see Subsection 5.2.5).

We naturally group these AF-learners relative to the algorithm they are derived from. Thus we obtain 33 AF-learners for the kNN algorithm, 15 AF-learners for the PART algorithm etc. (in Table 5.6 the number of AF-learners for each algorithm is shown). Altogether we obtain 99 AF-learners (99 equals to the sum of numbers in the last column of Table 5.6).

First, for each pair consisting of AF-learner (out of 99) and data set (out of 20), we calculate the value of the performance measure (in %). From now on, any such value will be called the *score*. In Figure 5.2, we recall and summarise how the score is computed relative to the performance measure, AF-learner, and data set. Let us also recall that scores for all AF-learners were computed under the same conditions (for all AF-learners precisely the same splits in the cross-validation process are used – in all ten repetitions).

Thus, for all 99 AF-learners, we calculate the vectors of 20 scores (for 20 data sets). Next, these scores are used to generate representative scores (vector of scores for 20 data sets) for each particular algorithm. Finally, these representative scores (vectors of scores) are used for presenting the final comparative scores, and statistical evaluation of the experimental results.

The following three *strategies* for generating the vector of representative scores are used:

1. the *def* strategy taking the vector of scores of *a priori* fixed default AF-learner (in the group corresponding to each particular algorithm),

2. the *opt* strategy taking the vector of scores of 'the best' (for the used data sets) AF-learner in the group,

3. the *max* strategy constructing the vector of scores of the best scores in the group for each data set.

Figure 5.3 illustrates these three strategies. For any algorithm used in the comparison, each of these three strategies transforms the scores (vectors of scores) of AF-learners for the algorithm into the vector of representative scores for the algorithm. In result, each of these three strategies transforms the 99×20 matrix of scores for 99 AF-learners and 20 data sets into the 10×20 matrix of representative scores for 10 algorithms. These matrices are used in the final comparisons (3 matrices for 3 strategies).

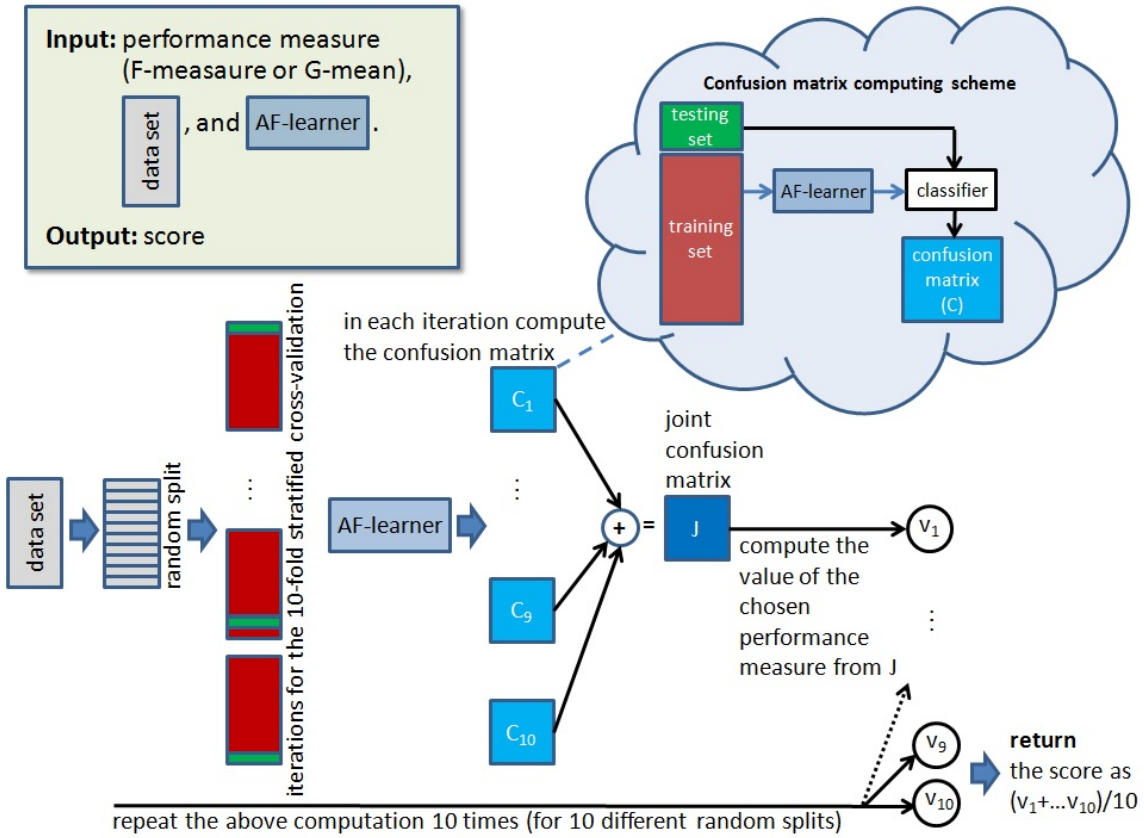


Figure 5.2: Illustration of computing of the score for any pair consisting of data set and AF-learner (for the chosen performance measure). 1) Data set is randomly split into 10 roughly equal parts so that in each fold, the distribution of classes is roughly the same as in the original data set. 2) In each iteration of the 10-fold stratified cross-validation, the confusion matrix is computed. 3) These matrices are added (simple matrix addition). 4) From the joint confusion matrix, the chosen performance measure is computed (this relates to the micro-average style of computing the performance measure; see Subsection 2.6.2). 5) These steps 1-4 are repeated 10 times for 10 different random splits, and the average of the obtained results is returned as the score. It should be noted that for all AF-learners the same splits are used. It can be thought as the computations are performed in parallel for all AF-learners (and in parallel scores for all AF-learners are returned).

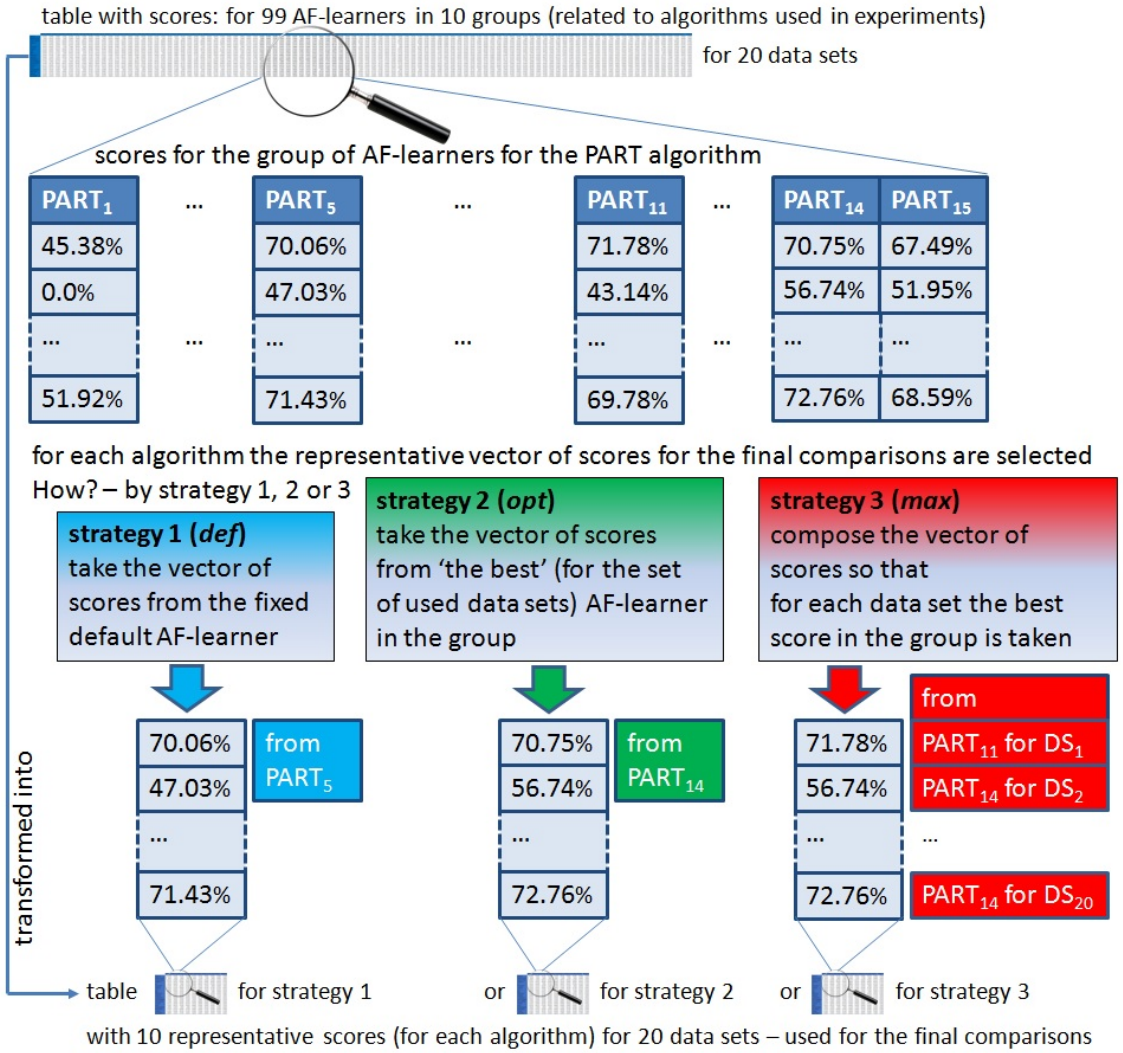


Figure 5.3: Illustration of the selection of the representative vector of scores for the final comparison for the three strategies. Partial scores and their transformations for the exemplary PART algorithm are shown. Analogously the transformations for other algorithms are performed. The meaning of 'the best' in strategy 2 (*opt*) will be explained later.

The whole process depends on the performance measure (fixed *a priori* by the user) used in the generation of the initial matrix. The experiments were performed separately for F-measure and G-mean (with 3 strategies for each of them; see Section 5.3). The choice of the measure obviously may influence the process of selection of the representative vector of scores. For example, the optimal AF-learner in the *opt* strategy depends, in particular, on the chosen performance measure. In Section 5.3, we sometimes add suffixes 'G' or 'F' to the names of the strategies to denote that the strategy uses G-mean or F-measure, respectively.

Details concerning each of these three strategies are explained in the following subsubsections. It is assumed for each strategy that the matrix with scores of all AF-learners is given. Such a matrix creates, in a sense, the input to all of these three strategies.

Def strategy

For each algorithm used in the experiments, we specify *a priori* the default AF-learner. In the *def* strategy, the most simple one, the scores (vector of scores) of the default AF-learner are used as the representative scores for the algorithm.

For each algorithm used in the experiments, we need only to describe how to set up one default AF-learner (independently of data sets). Information about the performance measure chosen by the user can be used only in RIONIDA, i.e. option for such setting is available only for this algorithm. Since for other used algorithms no such option is available, therefore for these algorithms, the default AF-learner is also independent of the chosen performance measure. For each algorithm used in the experiments, we define the default AF-learner by means of (i) the default configuration of the algorithm, and (ii) the default configuration of filters for the algorithm.

Certainly, the default configuration of a given algorithm is the combination of default options for this algorithm (default use of this algorithm). In practice, in most cases, using an algorithm with no options is its default use. The default values of non-binary options used in our experiments are given in Table 5.3 (for binary options their omission relates to their default use). Specifically, we use the following default configurations of algorithms: kNN `-K 1` (equivalent to kNN, i.e. using kNN without any option), PART `-C 0.25` (equivalent to PART), J48 `-C 0.25` (equivalent to J48), RIPPER, RISE, MODLEM, RIONA, BRACID. The default configuration of RIONIDA is described in the next paragraph. The only exception is the MODLEM-C algorithm in which the default setting is not specified within the algorithm. For this algorithm, we use as default the following setting MODLEM-C `-M 10` (strength multiplier for the minority class equal to 10). In fact, we use here the setting which was found as the optimal one in the *opt* strategy.

The default configuration of RIONIDA is RIONIDA `-T 0` (called in the thesis RIONIDA_G) or RIONIDA `-T 1` (called in the thesis RIONIDA_F) for the chosen performance measure G-mean or F-measure, respectively. It should be noted that the optimisation for the chosen performance measure is done using only the given training set. In the process of the (stratified) cross-validation it alters; thus, different optimal values of internal parameters of RIONIDA are found for different iterations. It is worthwhile to recall here Subsection 4.4.1 for an explanation of how RIONIDA

is optimised for a fixed performance measure.

As the default configuration of filters, we use `SMOTE+ENN` for the algorithms not dedicated to imbalanced data, and `Null-filter` for those dedicated to imbalanced data. In fact, in case of the algorithms not dedicated to imbalanced data, we use the combination of filters which usually led to the best performance of these algorithms in the experiments related to the *opt* strategy. This is consistent with the already reported results from [14].

As the default AF-learner, we use a combination of the default configuration of the algorithm and the default configuration of filters. Specifically, it is defined by one of the pairs: ⟨algorithm with default options, `SMOTE+ENN`⟩ for the algorithms not dedicated to imbalanced data, and ⟨algorithm with default options, `Null-filter`⟩ for the algorithms dedicated to imbalanced data.

Let us note that we can use in this strategy the previously computed scores for all AF-learners because for each algorithm its default AF-learner (as described above) is included in the set of AF-learners used in the experiments (see Subsection 5.2.5). Of course, this strategy uses only a small part of the given 99×20 matrix of scores.

Opt strategy

One could perform comparative experiments using only the default AF-learner. However, as it was already mentioned, this could be not satisfactory. The ideas presented in this and the following subsubsection provide an opportunity to make a comparative study taking into account many possible AF-learners for the used algorithms.

It is important to note that for the RIONIDA algorithm (and also for BRACID) selecting the optimal AF-learner was omitted. It was done intentionally to compare the default setting for the RIONIDA algorithm with the 'best' possible settings for other algorithms.

The main idea for the *opt* strategy is based on the selection of one, 'optimal' AF-learner for each algorithm. Then its scores are used as the representative scores for the algorithm. Below, we explain what means selecting the 'optimal' AF-learner. The intuition is that we want to select the AF-learner which will be the most competitive in the context of the Friedman statistical test used at the end of the comparative process. Since Friedman statistical test uses average ranks of the learning algorithm, it is this factor that is taken into account while selecting the optimal AF-learner.

As it was mentioned, the 99×20 matrix of scores (for 99 AF-learners and 20 data sets) is given. We need to assess for each AF-learner, how well it performs on all data sets on average. Thus we count average rank[5] (like in Friedman test; see Subsection 2.6.4) for each of the AF-learners. From each of the group of AF-learners, we select the optimal one with the lowest (optimal) rank in the group. This AF-learner is selected as the representative for the final comparison of algorithms. In result, the representative scores (vector of scores) for the algorithm are simply copied from the scores (vector of scores) of the selected optimal AF-learner.

[5]Basically, the scores are transformed into numbers $1, 2, \ldots, 99$ corresponding to the best, the second best, ..., the worst score. Then, average rank, i.e. the average of these numbers for all data sets, is calculated.

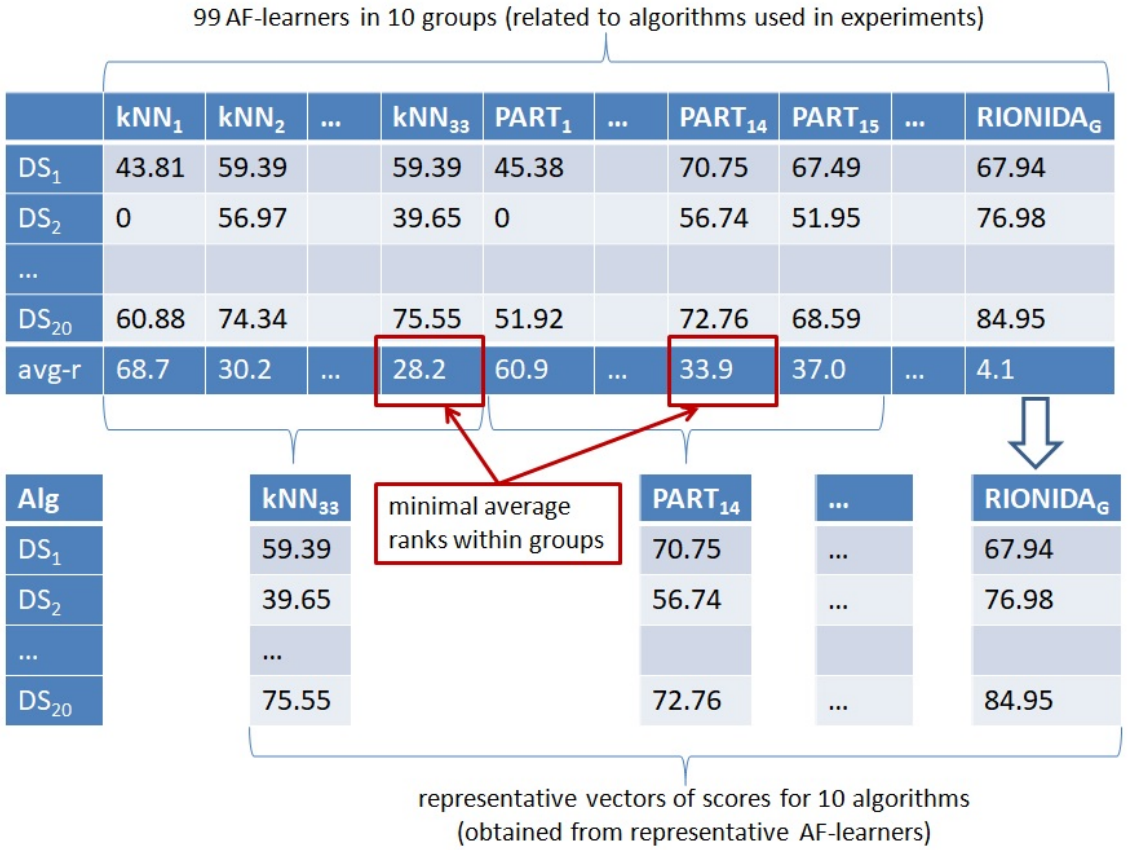


Figure 5.4: Illustration of the *opt* strategy. Scores, shown as numbers in rows $DS_1, \ldots, DS_{20}$ (different data sets) denote the value of the performance measure (in %) for different AF-learners and data sets. These scores were transformed into ranks, and the average ranks (avg-r) over all used data sets were computed (see Subsection 2.6.4 how avg-r is computed). From each group of AF-learners, the one with the minimal average rank (avg-r) within the specific group was selected as the (optimal) representative AF-learner. Scores (vector of scores) of this selected representative AF-learner were used for the final comparison of algorithms. For the algorithms with only one AF-learner (RIONIDA and BRACID), their scores were simply re-written for the final comparison of algorithms.

This strategy is illustrated in Figure 5.4 in case of G-mean performance measure (thus RIONIDA is set to $\text{RIONIDA}_{\text{G}}$). In the analysis of the figure, the readers are advised first to concentrate on the average ranks of AF-learners (avg-r). In this illustration, the optimal AF-learners selected during this strategy are indicated as kNN_{33} for the kNN algorithm, and PART_{14} for the PART algorithm.

Additionally, we present the results (which are discussed in Subsection 5.3.1 for the *opt* strategy) with some more details to allow the readers to better understand the used strategy. Table 5.8 presents the best three AF-learners in each group with their average ranks. Only the best AF-learner from each group was selected as the (optimal) representative AF-learner for the final comparisons. Other AF-learners are presented in the table only to point out other top candidates. For example, for the kNN algorithm, the following AF-learner was selected: ⟨kNN `-K 100 -X`, `SMOTE+ENN`⟩ (options for the kNN algorithm which automatically search for the optimal k between 1 and 100; for preprocessing the training set, the `SMOTE` filter and then `ENN` is used). This AF-learner is labelled in Figure 5.4 as kNN_{33}. For the PART algorithm, the following AF-learner was selected: ⟨PART `-U`, `SMOTE+ENN`⟩ (the PART algorithm with unpruned option; for preprocessing the training set, the `SMOTE` filter and then `ENN` is used). This AF-learner is labelled in Figure 5.4 as PART_{14}. For other algorithms, their optimal AF-learners can be found in Table 5.8 as bold items. Let us note that for both the BRACID and RIONIDA algorithms selecting the optimal AF-learner is unnecessary. In these cases default AF-learners are used, namely ⟨BRACID , `Null-filter`⟩ for BRACID, and ⟨$\text{RIONIDA}_{\text{F}}$, `Null-filter`⟩ or ⟨$\text{RIONIDA}_{\text{G}}$, `Null-filter`⟩ for the RIONIDA algorithm (depending on whether F-measure or G-mean was chosen as the performance measure of our interest).

Let us sum up these exemplary details and connect these results with the results used for the final comparison. Data sets referred to as DS_1, DS_2, and DS_{20} in Figure 5.4 in fact, indicate *abalone*, *balance-scale*, and *yeast* data sets, respectively. Thus, for the kNN algorithm, AF-learner ⟨kNN `-K 100 -X`, `SMOTE+ENN`⟩ was selected (with the following scores for different data sets: 59.39% for *abalone*, 39.65% for *balance-scale*, ..., and 75.55% for *yeast*). For the PART algorithm, AF-learner ⟨PART `-U`, `SMOTE+ENN`⟩ was selected (with the following scores for different data sets: 70.75% for *abalone*, 56.74% for *balance-scale*, ..., and 72.76% for *yeast*). These scores (obtained by the described strategy for G-mean measure) one can find in Table 5.9.

Interpretation and remarks on the *opt* strategy

The *opt* strategy somehow helps us to use the most competitive versions of algorithms in the final experiment (see Table 5.9). Since we performed it within all AF-learners (not only within groups of variations of the considered algorithms), this can help to find algorithms with their optimal AF-learners. These algorithms are expected to be the most competitive with RIONIDA.

It should be emphasised that in the presented process of searching for the optimal AF-learner for algorithm different from RIONIDA (and BRACID), the information from the test part of data sets is also used (see Subsubsection 'Remarks on the three strategies' on page 162). To be precise, specific AF-learners use only training sets. However, the selection of the optimal AF-learner is done using the average ranks obtained with the use of test part of data sets. It gives these algorithms an advantage

over RIONIDA (and also BRACID) in the process of the performance comparison. This is because RIONIDA (and also BRACID) uses only one fixed default AF-learner. Let us assume that someone is tuning options and filters for the algorithm using the fixed sets of AF-learners relative to the selected data sets. Then, most likely,[6] one cannot achieve a better result (in terms of average rank) than achieved with this optimal AF-learner for the algorithm.

Max strategy

Here, we present the most competitive strategy (of the three presented) for selecting the representative scores for the algorithm. The main idea of the *max* strategy is to select for any considered algorithm the best score separately for each data set out of the scores of AF-learners in the group corresponding to the algorithm.

As it was mentioned, the 99×20 matrix of scores (for 99 AF-learners and 20 data sets) is given. For each data set from each group of the algorithm, the maximal score is selected. This score is used for the final comparison of algorithms for this particular data set. For the algorithms with one AF-learner (RIONIDA and BRACID), their scores are simply re-written for the final comparison of algorithms.

In the previous subsubsection, a single (optimal) AF-learner was selected for each algorithm. Then its scores were used as the representative scores for the algorithm. This strategy can be seen as selecting the optimal AF-learner separately for each data set.

The *max* strategy is illustrated in Figure 5.5 in case of G-mean performance measure (thus RIONIDA is set to $\text{RIONIDA}_{\text{G}}$). The (exemplary) optimal AF-learner for the kNN algorithm selected during this strategy are as follows: kNN_8 for data set DS_1; kNN_3 for data set DS_2 and kNN_8 for data set DS_{20}. The optimal AF-learner for the PART algorithm are as follows: PART_{11} for data set DS_1; PART_{14} for data set DS_2 and PART_{14} for data set DS_{20}.

To allow the readers to better understand the used idea, we expand the presented results by providing some more details. The AF-learner referred in Figure 5.5 as kNN_3 indicates the AF-learner ⟨kNN `-K 1`, `SMOTE`⟩ (options for the kNN algorithm with fixed $k = 1$; for preprocessing the training set, the `SMOTE` filter is used). The AF-learner referred as kNN_8 indicates the AF-learner ⟨kNN `-K 2`, `SMOTE+ENN`⟩ (options for the kNN algorithm with fixed $k = 2$; for preprocessing the training set, the `SMOTE` filter and then the `ENN` filter is used). Data sets referred as DS_1, DS_2, and DS_{20} indicate *abalone*, *balance-scale*, and *yeast* data sets, respectively. In particular, for the kNN algorithm, different AF-learners were selected (with the above-described strategy) for different data sets: ⟨kNN `-K 2`, `SMOTE+ENN`⟩ for *abalone* (with the score 63.46%), ⟨kNN `-K 1`, `SMOTE`⟩ for *balance-scale* (with the score 65.34%), and ⟨kNN `-K 2`, `SMOTE+ENN`⟩ for *yeast* (with the score 79.83%). All these scores can be found in Table 5.10, which will be analysed in the next section. The scores in the

[6] The presented process is a kind of heuristic. In this way, what we call optimal is, in fact, a pseudo-optimal AF-learner, i.e. another AF-learner can possibly achieve better average rank than the selected one in this heuristic as the optimal AF-learner. This relates to the fact that the average rank of the considered algorithm depends not only on its performance but also on the other algorithms used in comparisons. Here, we consider all possible AF-learners. Later we use only AF-learners selected in this step.

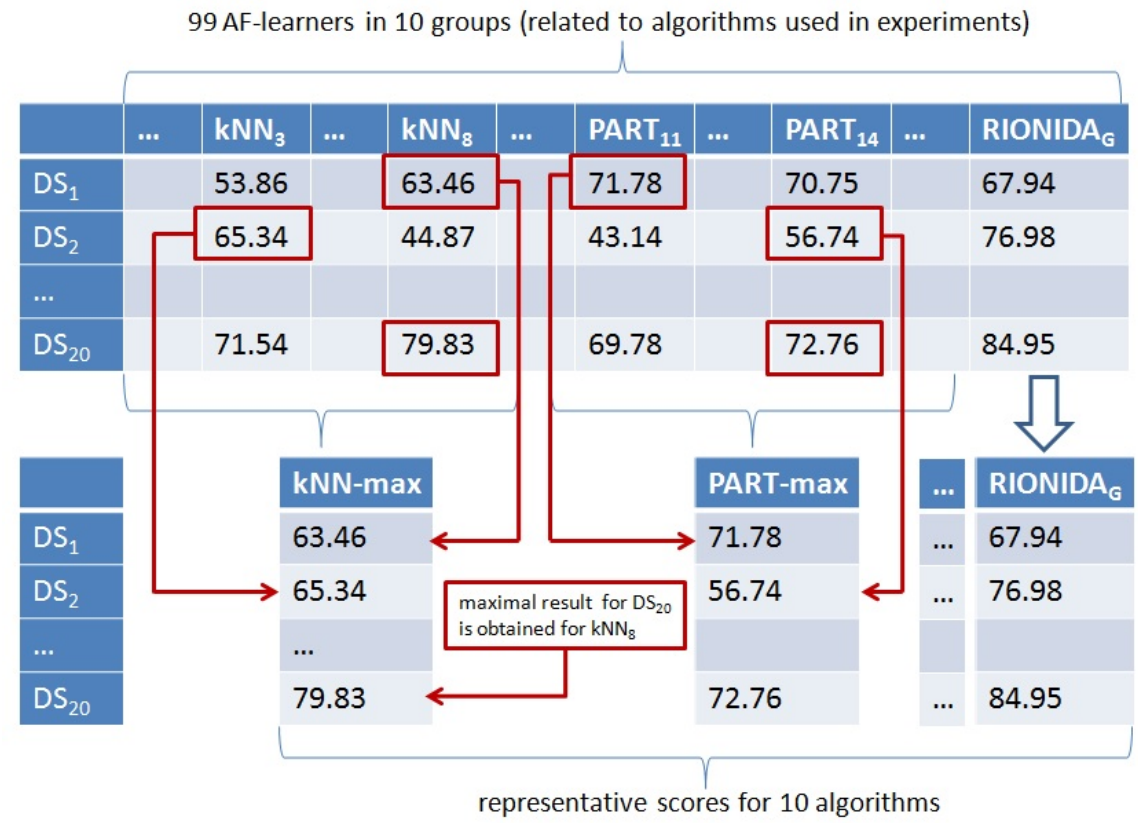


Figure 5.5: Illustration of the *max* strategy. First, experiments were performed jointly for all algorithms with all selected AF-learners. Scores, shown as numbers in rows $DS_1, \ldots, DS_{20}$ (different data sets) denote the value of the performance measure (in %) for different algorithms and data sets. Then for each data set from each group of AF-learners, the maximal score was selected. These maximal scores are marked with red rectangles. These scores were used for the final comparison of algorithms for particular data sets. For the algorithms with one AF-learner (RIONIDA and BRACID), their scores were simply re-written for the final comparison of algorithms.

table come from the described strategy. Above, we explained how these scores were obtained. However, when the final results are presented in the next section, e.g. in Table 5.10, no such detailed information will be given.

Interpretation and remarks on the *max* strategy

Let us assume that for each algorithm used in the experiments, one constructs a meta-learning algorithm that learns the optimal AF-learner for a given training sample (using some validation scheme). For each algorithm, the *max* strategy provides an (upper) approximation of the scores of such meta-learning algorithm.

In fact, with the above interpretation, the presented strategy relates to using additionally the test parts of data set (for the optimal AF-learner selection). To be precise, specific AF-learners use only training sets. However, the selection of the optimal AF-learner is done using the scores obtained with the use of test part of data sets (see Subsubsection 'Remarks on the three strategies' below).

It should be noted that the kNN algorithm used in the experiments has internally implemented learning of its parameter k and we use such a functionality (option `-X`). However, by using the *max* strategy for the kNN algorithm, we still give an advantage for this algorithm (we emulate the possibility of using even better meta-learning scheme) in comparison with RIONIDA.

Remarks on the three strategies

Out of the three presented strategies, the *def* strategy is the most straightforward. It is commonly used in the literature for comparative experiments (default use of the algorithm is applied; possibly preceded with some fixed filter).

Two remaining strategies are more compound and return the scores, which potentially could be obtained by algorithms in experiments under some strong assumptions. Let us concentrate on the aforementioned interpretations for the *opt* and *max* strategies. In these interpretations, the *opt* strategy (*a posteriori*) tunes the options and filters (relative to the used data sets), and the *max* strategy 'learns' options and filters on the meta-learning level[7].

In such a case, the experiments could seem as improperly prepared since we set the optimal AF-learner for each algorithm using the information included in the test part of data sets. However, by arranging the experiment in this way we give an advantage for algorithms with more than one AF-learner over RIONIDA (in fact also over BRACID, which also has no variability in the set of AF-learners).

The *opt* strategy can be seen as comparing RIONIDA with other algorithms under consideration using their optimal (in terms of average ranks) AF-learners (out of the presented ones). These optimal AF-learners were selected taking into account the used data sets (for all algorithms excluding RIONIDA). In particular, if all the learning algorithms were set by chance with other options and filters than the default ones, then one could expect that the obtained results were lower (for other algorithms than RIONIDA) than the presented ones. If for this strategy, RIONIDA could be

[7]It should be noted that normally meta-learning scheme would select different optimal options and filters for different splits in the cross-validation process. However, here the common optimal options are used for all splits.

shown to be statistically better than some algorithms, then this should be perceived as a strong result in favour of the RIONIDA algorithm.

The *max* strategy can be seen as comparing RIONIDA with other algorithms under consideration using the upper bound of scores (for individual data sets) obtained when meta-learning for the selection of AF-learners was implemented for algorithms other than RIONIDA. In particular, if all learning algorithms were supported with the possibility of learning of the optimal AF-learners (using only the training data), one could expect that the obtained scores were lower (for other algorithms than RIONIDA) than the presented ones in the comparisons for the *max* strategy. If for this strategy, RIONIDA could be shown to be statistically better than some algorithms, this should be perceived as a very strong result in favour of the RIONIDA algorithm.

5.3 Comparison of RIONIDA with the selected state-of-the-art algorithms

Let us recall that RIONIDA was constructed for data analysis in a possibly wide range of application domains. In this section, we will try to answer the question whether the presented algorithm can be evaluated as 'better' than the other algorithms used in the comparison (with subject to all remarks in Section 2.6).

An important step in the process of evaluation of learning algorithms is related to statistical tests (see Section 2.6). Generally, we use the Finner statistical test (after the Friedman test), characterised by the relatively high power. However, we also use the Nemenyi statistical test due to its clear graphical interpretation. It is used whenever the Finner test shows that RIONIDA is significantly better than all other algorithms used in the comparison, and simultaneously the Nemenyi test shows the same. In such a case, we can present the results in a more compact graphical form. Then by using the Nemenyi test, additionally the comparison of other pairs of algorithms is given. However, without focusing on such comparisons, we only very briefly discuss the related issues.

As it was mentioned previously, two general groups of the comparative experiments were performed corresponding to the chosen performance measure: G-mean and F-measure. The results for these measures are shown in the following Subsections 5.3.1 and 5.3.2, respectively.

5.3.1 Comparison of algorithms for G-mean

In this section, we assume that the performance measure we are interested in is G-mean. Thus, the particular parameter of RIONIDA is set to optimise G-mean. The algorithm with this setting is called $\text{RIONIDA}_{\text{G}}$. In this section, the representative scores for RIONIDA are fixed, i.e. these are simply scores for $\text{RIONIDA}_{\text{G}}$. In the following subsections, we permanently underline this fact that for $\text{RIONIDA}_{\text{G}}$ (and BRACID) one default AF-learner was used. Thus, irrespective of the used strategy, the scores for $\text{RIONIDA}_{\text{G}}$ (and BRACID) are the same for the three considered strategies. For other algorithms used in the comparative experiments,

their representative scores are selected relative to the G-mean measure and the data sets used in the experiments (and certainly to the used strategy; see Subsection 5.2.6).

We present the results of comparative experiments for three strategies:

1. the *defG* strategy (the *def* strategy for G-mean),
2. the *optG* strategy (the *opt* strategy for G-mean), and
3. the *maxG* strategy (the *max* strategy for G-mean) (see Subsection 5.2.6).

If we say something about an algorithm (e.g. kNN) in the context of experiments, we relate it to the representative scores obtained with the considered strategy for the given algorithm.

Def strategy for G-mean (*defG*)

In this step, we compare algorithms using their default AF-learners (as described in Subsection 5.2.6 for the *def* strategy). Specifically, we compare the following AF-learners: ⟨kNN `-K 1`, `SMOTE+ENN`⟩, ⟨PART `-C 0.25`, `SMOTE+ENN`⟩, ⟨J48 `-C 0.25`, `SMOTE+ENN`⟩, ⟨RIPPER, `SMOTE+ENN`⟩, ⟨RISE, `SMOTE+ENN`⟩, ⟨MODLEM, `SMOTE+ENN`⟩, ⟨MODLEM `-M 10`, `Null-filter`⟩, ⟨RIONA, `SMOTE+ENN`⟩, ⟨BRACID, `Null-filter`⟩, ⟨RIONIDA$_{\mathrm{G}}$, `Null-filter`⟩.

In Table 5.7, for each learning algorithm, the representative scores of G-mean (for *defG*) for all used data sets are given. The RIONIDA algorithm was set to optimise the G-mean measure, i.e. RIONIDA$_{\mathrm{G}}$ was used; hence, RIONIDA$_{\mathrm{G}}$ appears in the table instead of RIONIDA.

Then the algorithms were ranked for each data set (see Subsection 2.6.4 for details). For illustration, for five algorithms on the right in Table 5.7, we present (in parentheses) their ranks. It should be noted that the later used Friedman statistical test (and post-hoc tests) makes use only of these ranks (not specific values of scores).

One can see from Table 5.7 that in most cases, the RIONIDA algorithm achieves the best score: for 20 data sets, 15 times it wins with all other algorithms (in these cases RIONIDA has the rank equal to 1), and once (for *new-thyroid*) its score is equal to the other algorithm (namely, RIONA) with the best score (in this case RIONIDA has rank equal to 1.5). In situations when it loses with an algorithm, the difference between the best score and the score of RIONIDA is: once about 5%, once about 1%, and twice below 0.5%. For these cases, RIONIDA has the following ranks: 4 (for *abalone*), 3 (for *ionosphere*), and 2 (for *breast-w* and *vehicle*).

Next, the average rank for each algorithm (more precisely for each AF-learner consisting of an algorithm with default parameters and default filter) was computed. In the third line from below of Table 5.7, the average ranks (for all algorithms) are presented. The lower the average rank is, the better learning algorithm is. The average of all ranks for RIONIDA gives the result 1.375, which is the best outcome. The difference between the average rank of the RIONIDA algorithm and the second-lowest average rank (4.6 for BRACID) is relatively high (3.225).

By using these average ranks (and particular ranks), the Friedman statistic was computed[8] returning the result 55.814. With 10 algorithms, this statistic follows

[8]Let us recall that this test takes into account the variations in the ranks of algorithms.

Table 5.7: The values of G-mean (in %) for different algorithms and different data sets, for *defG*. The RIONIDA algorithm was set to optimise the G-mean measure (i.e. $\text{RIONIDA}_{\text{G}}$ was used). For each data set, the best-obtained score is shown in bold. Also, for illustration, for five algorithms on the right (including $\text{RIONIDA}_{\text{G}}$), ranks for these algorithms and different data sets are shown (in parentheses). At the bottom are shown: (i) average rank for each algorithm, (ii) important outcomes of the Friedman statistical test (Friedman statistic, degrees of freedom, and p-value), and (iii) adjusted p-values (APV) with the Finner post-hoc test using RIONIDA as the control algorithm.

	The vectors of representative scores for different learning algorithms generated with the *defG* strategy									
Data set	**kNN**	**PART**	**J48**	**RIPPER**	**RISE**	**MODLEM**	**MODLEM-C**	**RIONA**	**BRACID**	**$\text{RIONIDA}_{\text{G}}$**
abalone	59.39	70.06	70.36	**73.05**	60.03	65.54 (6)	55.06 (10)	59.91 (8)	65.80 (5)	67.94 (4)
balance-scale	56.97	47.03	22.70	13.58	40.97	12.90 (9)	2.78 (10)	33.26 (6)	58.68 (2)	**76.98 (1)**
breast-cancer	56.64	53.99	54.23	53.53	58.26	56.04 (7)	58.34 (2)	56.92 (5)	58.17 (4)	**64.98 (1)**
breast-w	97.36	96.28	95.79	95.99	96.89	96.16 (7)	94.89 (10)	**97.81 (1)**	96.91 (4)	97.53 (2)
car	83.23	86.68	87.40	80.08	75.89	82.51 (7)	89.23 (2)	80.37 (8)	87.47 (3)	**96.74 (1)**
cleveland	63.83	63.77	66.48	69.68	59.43	63.34 (7)	35.61 (10)	65.27 (4)	62.89 (8)	**76.38 (1)**
credit-g	65.66	66.30	66.17	65.59	65.27	65.57 (8)	66.78 (2)	66.35 (3)	62.27 (10)	**69.90 (1)**
ecoli	86.82	84.82	84.11	86.36	85.59	84.15 (8)	67.65 (10)	86.68 (3)	84.42 (7)	**88.82 (1)**
glass	62.75	64.30	67.89	57.00	54.69	63.16 (5)	47.64 (9)	66.80 (3)	39.90 (10)	**69.26 (1)**
haberman	59.76	61.64	62.41	61.90	60.35	62.64 (2)	57.10 (10)	59.85 (7)	59.55 (9)	**65.40 (1)**
hepatitis	74.94	67.56	66.78	65.82	71.16	71.71 (5)	67.55 (8)	73.46 (4)	77.11 (2)	**79.00 (1)**
ionosphere	89.91	86.88	85.64	84.27	**91.91**	85.93 (8)	89.55 (6)	90.37 (4)	91.42 (2)	90.89 (3)
mammography	73.48	72.59	71.73	73.37	73.18	70.68 (9)	68.74 (10)	74.17 (3)	85.41 (2)	**89.70 (1)**
new-thyroid	98.71	95.13	95.05	95.16	97.73	94.36 (9)	92.92 (10)	**98.93 (1.5)**	98.69 (4)	**98.93 (1.5)**
nursery	89.99	97.12	87.04	84.43	83.82	97.05 (4)	99.80 (2)	88.73 (7)	96.58 (5)	**99.90 (1)**
pima	66.75	67.07	67.47	68.42	67.99	65.41 (10)	69.96 (3)	66.54 (9)	71.28 (2)	**72.87 (1)**
postoperative	38.32	36.03	37.33	33.91	36.84	34.75 (8)	40.21 (3)	34.02 (9)	42.49 (2)	**43.66 (1)**
transfusion	62.76	61.89	63.22	64.38	63.11	62.31 (8)	57.57 (10)	63.34 (4)	64.39 (2)	**67.64 (1)**
vehicle	93.27	93.51	93.03	93.06	92.59	93.67 (5)	**95.45 (1)**	94.47 (3)	93.82 (4)	95.10 (2)
yeast	74.34	71.43	70.08	73.60	68.78	64.55 (9)	46.95 (10)	75.92 (2)	72.38 (5)	**84.95 (1)**
average rank	5.2	5.9	6.3	6.45	6.5	7.05	6.9	4.725	4.6	**1.375**
Friedman test	Friedman's chi-squared = 55.814, df = 9, p-value = $8.52 \cdot 10^{-9}$									
APV Finner	0.00008	$< 10^{-5}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-7}$	$< 10^{-7}$	0.00053	0.00076	control

the Chi-square distribution with $df = 9$ (degrees of freedom). The obtained value exceeds the critical value for the Chi-square distribution (equal to 16.92 for $\alpha = 0.05$, i.e. the significance level used in the thesis, and $df = 9$). As we mentioned, the more informative it is to use the p-value. The obtained p-value is equal to $8.52 \cdot 10^{-9}$, which is much smaller than $\alpha = 0.05$. Anyway, we can safely reject the null hypothesis that all the algorithms perform equally well. This information is necessary to make a more informative step, i.e. post-hoc test. In the discussed table (and the next ones), we present the most important outcomes of the Friedman statistical test in the second line from below.

The best result (average rank) is achieved by RIONIDA$_{\mathrm{G}}$. The post-hoc test is used to detect whether the differences between this and the other learning algorithms are statistically significant. As it was mentioned, the Finner statistical test was used to compare all learning algorithms with the selected control one (in our case RIONIDA).

The adjusted p-values for each of the algorithm for the Finner statistical test (with the RIONIDA$_{\mathrm{G}}$ algorithm set as the control one) are the following: kNN $8.31487 \cdot 10^{-5}$; PART $3.431262 \cdot 10^{-6}$; J48 $4.841227 \cdot 10^{-7}$; RIPPER $2.596844 \cdot 10^{-7}$; RISE $2.596844 \cdot 10^{-7}$; MODLEM $2.770859 \cdot 10^{-8}$; MODLEM-C $3.552973 \cdot 10^{-8}$; RIONA 0.0005254441; BRACID 0.0007560509. The essential information for this test is whether each particular p-value is below (or even well below) 0.05 (5%)[9]. The smaller the p-value, the stronger is the evidence that the difference between RIONIDA and another considered algorithm (corresponding to the p-value) is statistically significant (see Subsection 2.6.4).

For all learning algorithms used in comparisons with RIONIDA$_{\mathrm{G}}$, the corresponding p-value is (much) smaller than $\alpha = 0.05$. Thus, we can (confidently) reject all the null hypotheses corresponding to the algorithms used in the comparison (at 0.05 level of significance). In other words, RIONIDA$_{\mathrm{G}}$ is significantly better than any other learning algorithm (with default AF-learners) relative to the G-mean performance measure.

In the discussed table (and the next ones) we present p-values rounded to 5 decimal places. In case the p-value is smaller than 10^{-5}, only its order of magnitude is indicated. For example, for the PART algorithm, it is shown that p-value is smaller than 10^{-5}.

Additionally, we also used the Nemenyi statistical test (for multiple comparisons) due to its clear graphical interpretation. Figure 5.6 shows the critical difference plot for the Nemenyi statistical test in this case. The line in the diagram with integer numbers is the axis on which we plot the average ranks of learning algorithms (to be precise default AF-learner for the learning algorithm). The diagram should be read in such a way that the better the learning algorithm is, the more it is to the left. The horizontal interval marked as CD is the critical difference in the Nemenyi statistical test. The performances of two algorithms are significantly different (with the level of significance at 0.05) according to this test if the corresponding average ranks differ by at least the critical difference. This fact is also marked in the diagram. We connect the groups of algorithms that are not significantly different (according to this test)

[9] If so, it means that differences between the newly presented algorithm and other algorithm are statistically significant. If not, it means that no statistical difference could be detected with this post-hoc test (the results could be statistically different or not – we still do not know the correct answer).

with a bold horizontal line (the one below the axis).

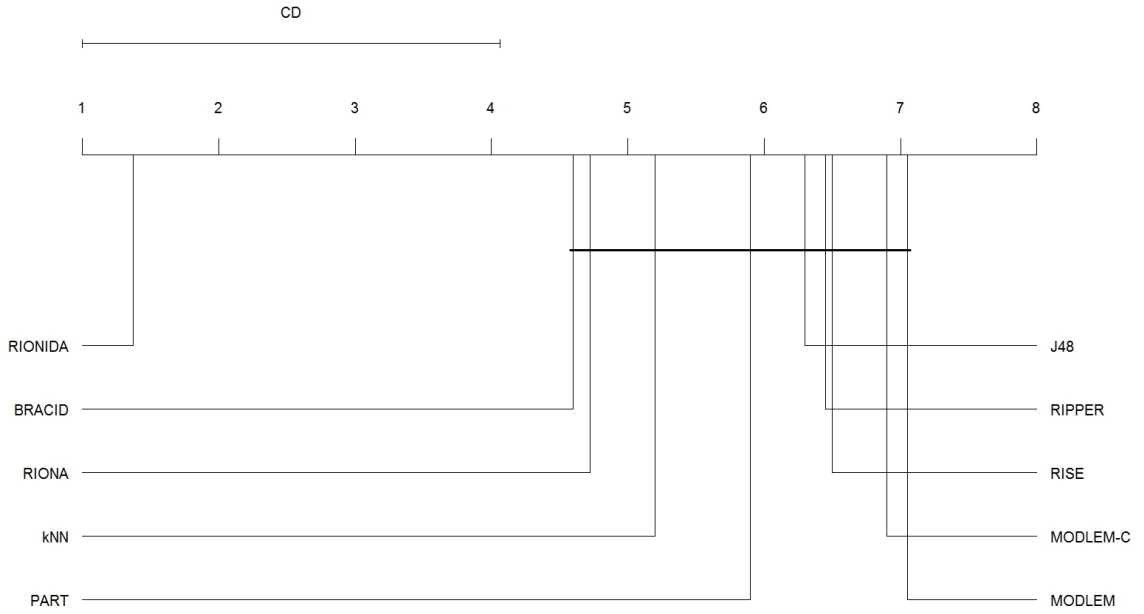


Figure 5.6: Comparison of G-mean for all algorithms used in the comparison (with the *defG* strategy) against each other with the Nemenyi statistical test. Groups of algorithms that are not significantly different (with the level of significance at 0.05) are connected.

This test, although conservative, shows that RIONIDA$_{\mathrm{G}}$ is significantly better (with the level of significance at 0.05) than all other algorithms used in the comparison. The difference of average ranks between RIONIDA$_{\mathrm{G}}$ and other algorithms also seems quite high in comparison to the differences between other algorithms compared. From this diagram, one can also see that all other algorithms are not statistically different according to the Nemenyi test. However, it is not the intention of the author to interpret this fact. One needs to remember that this test is very conservative and may show no differences in a situation when they actually exist. What we want to underline is that even such conservative test shows the difference between RIONIDA and any other algorithm used in the comparison.

The second best algorithm in terms of average ranks is the BRACID algorithm. The third one is RIONA (with a proper filter) and is slightly worse (in terms of average ranks) than BRACID. In particular, the Nemenyi statistical test shows that BRACID and RIONA are not statistically different. In particular, it should also be noted that RIONIDA is significantly better than the RIONA algorithm. This justifies that the changes made in RIONA were essential to deal with imbalanced data.

It should be noted that in this subsubsection, the explanations were given with many details. In the following considerations, we will omit many of them and give only the conclusions based on the obtained p-values for particular statistical tests.

Opt strategy for G-mean (*optG*)

Here, we will try to answer the following question: Can the general situation (with scores and ranks of algorithms) and conclusions described in the previous

subsubsection change if some other than default parameters and type of filter are selected (if possible) for algorithms (except RIONIDA)? To answer this question, for each algorithm were selected: (i) its optimal parameters, and (ii) the optimal type of filter regarding all data sets used in the comparison and the fact that G-mean is the performance measure. Here, 'optimal' relates to the *opt* strategy described in Subsection 5.2.6 and the G-mean performance measure (such strategy was called earlier more specifically *optG* strategy).

In Table 5.8, the optimal (i.e. with the lowest average rank) AF-learners for the *optG* strategy are presented. Also, two other top candidates are pointed out in the table (yet not used in the final comparison). For each learning algorithm, the (optimal) representative AF-learner is marked in bold. The scores of these representative AF-learners are used in the final comparison presented below. If we compare these selected optimal AF-learners with the default AF-learners (used in the *defG* strategy presented above) the differences in settings (options or filters) can be observed for the following algorithms: kNN, PART, J48, MODLEM. For the others, the used AF-learners are the same in both strategies (*defG* and *optG*).

Table 5.8: The three best AF-learners (with the lowest average ranks shown in parentheses) selected by the *optG* strategy (for each algorithm). The best AF-learners are shown in bold and are used in the final comparison (for the *optG* strategy).

algorithm	three best AF-learners in each group
kNN	⟨**kNN** `-K 100 -X`, `SMOTE+ENN`⟩ **(28.2)**, ⟨kNN `-K 2`, `SMOTE`⟩ (28.9), ⟨kNN `-K 1`, `SMOTE+ENN`⟩ (30.2)
PART	⟨**PART** `-U`, `SMOTE+ENN`⟩ **(33.9)**, ⟨PART `-C 0.5`, `SMOTE+ENN`⟩ (36.1), ⟨PART `-C 0.25`, `SMOTE+ENN`⟩ (36.4)
J48	⟨**J48** `-C 0.5`, `SMOTE+ENN`⟩ **(38.5)**, ⟨J48 `-C 0.1`, `SMOTE+ENN`⟩ (38.8), ⟨J48 `-A`, `SMOTE+ENN`⟩ (39)
RIPPER	⟨**RIPPER** , `SMOTE+ENN`⟩ **(41.9)**, ⟨RIPPER `-E`, `SMOTE+ENN`⟩ (43.1), ⟨RIPPER `-E -P`, `SMOTE+ENN`⟩ (44)
RISE	⟨**RISE** , `SMOTE+ENN`⟩ **(38.3)**, ⟨RISE , `SMOTE`⟩ (50.7), ⟨RISE , `Null-filter`⟩ (66.8)
MODLEM	⟨**MODLEM** , `SMOTE`⟩ **(41.9)**, ⟨MODLEM , `SMOTE+ENN`⟩ (43.1), ⟨MODLEM , `Null-filter`⟩ (75.1)
MODLEM-C	⟨**MODLEM** `-M 10`, `Null-filter`⟩ **(48.3)**, ⟨MODLEM `-M 9`, `Null-filter`⟩ (49.2), ⟨MODLEM `-M 7`, `Null-filter`⟩ (49.4)
RIONA	⟨**RIONA** , `SMOTE+ENN`⟩ **(29.4)**, ⟨RIONA , `SMOTE`⟩ (36.1), ⟨RIONA , `Null-filter`⟩ (70.4)
BRACID	⟨**BRACID** , `Null-filter`⟩ **(24.8)**
RIONIDA	⟨**RIONIDA$_{\mathbf{G}}$** , `Null-filter`⟩ **(4.1)**

In Table 5.9, for each learning algorithm, the representative scores (for the *optG* strategy) for all used data sets are given. Analogously as in the table related to the *defG* strategy, RIONIDA$_{\mathrm{G}}$ appears in Table 5.9 instead of RIONIDA (which means

that RIONIDA was set to optimise the G-mean measure). In the table (and in its caption), we underline the fact that for $\text{RIONIDA}_{\text{G}}$ (and BRACID), the only one fixed, default AF-learner was used (i.e. the algorithm was preceded with no filter and used with its default parameters). Let us recall that by such choice we give an advantage for the algorithms with more than one AF-learner over RIONIDA (and BRACID). The readers should keep this fact in mind when reading the presentation of results below.

One can see from Table 5.9 that in most cases, the RIONIDA algorithm achieves the best score: for 20 data sets, 15 times it wins with all other algorithms, and once its score is equal to the other algorithm with the best score. In situations when it loses with an algorithm, the difference between the best score and the score of RIONIDA is: once about 5%, once about 1%, and twice below 0.5%. For these cases, RIONIDA has the following ranks: 4 (for *abalone* and *ionosphere*), 3 (for *vehicle*), and 2 (for *breast-w*). These observations are similar to those reported for the *defG* strategy. However, in particular, ranks for $\text{RIONIDA}_{\text{G}}$ are slightly worse than previously (for *ionosphere* and *vehicle*).

In the third line from below of Table 5.9, the average ranks (for all algorithms) are presented. The results mentioned above again give the best outcome of the average rank for RIONIDA (1.475). In comparison to the *defG* strategy, this result is worse only by 0.1. The difference between the average rank of the RIONIDA algorithm and the second-lowest average rank (4.8 for BRACID) is relatively high (3.325).

Let us now concentrate on the outcomes of the Friedman statistical test presented in the discussed table. One can see that this test gave very small p-value (less than 10^{-7}). This means that there exist statistical differences among compared algorithms.

As a consequence, one could perform the post-hoc Finner test for comparisons with the control algorithm (and the Nemenyi statistical test for multiple comparisons – performed additionally; see below). In the last line of the table, adjusted p-values of the post-hoc Finner test with RIONIDA as the control algorithm are presented.

The values in the table indicate that the RIONIDA algorithm is significantly better than any other algorithm (with the level of significance at 0.05). This is a quite astonishing result. Moreover, the highest p-value reaches 0.00051 (for BRACID), which is much less than 0.05 (less than 10^{-3}). The second and the third highest p-values are for the RIONA and kNN algorithms (with their optimal AF-learners), respectively and are also less than 10^{-3}.

As in the previous subsubsection, we also used the Nemenyi statistical test (for multiple comparisons) due to its clear graphical interpretation. Figure 5.7 shows the critical difference plot for the Nemenyi test in this case. One can see that $\text{RIONIDA}_{\text{G}}$ is significantly better (with the level of significance at 0.05) from all other algorithms. The difference in average ranks between $\text{RIONIDA}_{\text{G}}$ and other algorithms also seems relatively high in comparison to differences between other algorithms compared. Generally, the plot is similar to the one presented in the previous subsubsection (see Figure 5.6). Other conclusions from the previous subsubsection hold too.

One of the differences is in the position of the kNN algorithm, which is closer to the RIONA algorithm (in terms of average rank). However, one should keep in mind that due to many considered AF-learners for kNN, it has an advantage over BRACID (and RIONIDA). The selected optimal AF-learner for the kNN algorithm,

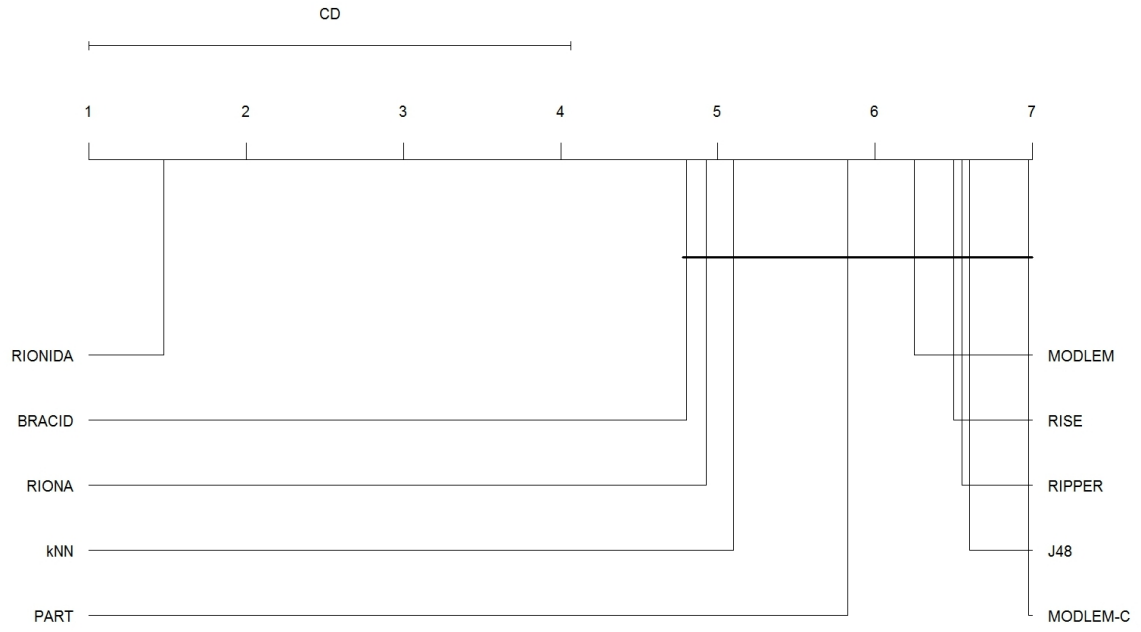


Figure 5.7: Comparison of G-mean for all algorithms used in the comparison (with the *optG* strategy) against each other with the Nemenyi statistical test. Groups of algorithms that are not significantly different (with the level of significance at 0.05) are connected.

Table 5.9: The values of G-mean (in %) for different algorithms and different data sets, for the *optG* strategy (additionally for RIONIDA ranks are shown in parentheses). The RIONIDA algorithm was set to optimise the G-mean measure (i.e. $\text{RIONIDA}_{\text{G}}$ was used). It should be noted that for both $\text{RIONIDA}_{\text{G}}$ and BRACID, one default AF-learner was used. For the other learning algorithms, the optimal AF-learner was selected using the *optG* strategy (and then it was used the same for all data sets). For each data set, the best-obtained score is shown in bold. At the bottom are shown: (i) average rank for each algorithm, (ii) important outcomes of the Friedman statistical test (Friedman statistic, degrees of freedom, and p-value), and (iii) adjusted p-values (APV) with the Finner post-hoc test using RIONIDA as the control algorithm.

	The vectors of representative scores for different learning algorithms generated with the *optG* strategy (for $\text{RIONIDA}_{\text{G}}$ and BRACID, one default AF-learner was used – their scores are the same as in the *defG* strategy)									
Data set	**kNN**	**PART**	**J48**	**RIPPER**	**RISE**	**MODLEM**	**MODLEM-C**	**RIONA**	**BRACID**	**$\text{RIONIDA}_{\text{G}}$**
abalone	59.39	70.75	70.04	**73.05**	60.03	61.54	55.06	59.91	65.80	67.94 (4)
balance-scale	39.65	56.74	22.46	13.58	40.97	0.00	2.78	33.26	58.68	**76.98 (1)**
breast-cancer	56.64	55.50	54.95	53.53	58.26	58.87	58.34	56.92	58.17	**64.98 (1)**
breast-w	97.37	96.35	95.68	95.99	96.89	95.84	94.89	**97.81**	96.91	97.53 (2)
car	86.23	85.26	86.86	80.08	75.89	88.06	89.23	80.37	87.47	**96.74 (1)**
cleveland	64.42	67.19	66.39	69.68	59.43	45.84	35.61	65.27	62.89	**76.38 (1)**
credit-g	65.66	66.78	66.13	65.59	65.27	65.65	66.78	66.35	62.27	**69.90 (1)**
ecoli	86.70	85.14	84.16	86.36	85.59	77.35	67.65	86.68	84.42	**88.82 (1)**
glass	65.71	60.68	66.79	57.00	54.69	63.82	47.64	66.80	39.90	**69.26 (1)**
haberman	59.75	61.25	62.52	61.90	60.35	57.32	57.10	59.85	59.55	**65.40 (1)**
hepatitis	76.70	69.53	68.08	65.82	71.16	73.42	67.55	73.46	77.11	**79.00 (1)**
ionosphere	**91.93**	87.11	85.32	84.27	91.91	85.54	89.55	90.37	91.42	90.89 (4)
mammography	73.45	72.95	71.76	73.37	73.18	79.96	68.74	74.17	85.41	**89.70 (1)**
new-thyroid	98.85	95.45	95.05	95.16	97.73	95.33	92.92	**98.93**	98.69	**98.93 (1.5)**
nursery	89.02	97.43	88.72	84.43	83.82	99.63	99.80	88.73	96.58	**99.90 (1)**
pima	66.75	66.56	67.64	68.42	67.99	69.45	69.96	66.54	71.28	**72.87 (1)**
postoperative	38.32	33.08	36.53	33.91	36.84	34.14	40.21	34.02	42.49	**43.66 (1)**
transfusion	62.76	61.73	63.21	64.38	63.11	59.02	57.57	63.34	64.39	**67.64 (1)**
vehicle	93.27	94.02	93.05	93.06	92.59	95.24	**95.45**	94.47	93.82	95.10 (3)
yeast	75.55	72.76	70.06	73.60	68.78	55.77	46.95	75.92	72.38	**84.95 (1)**
average rank	5.1	5.825	6.6	6.55	6.5	6.25	6.975	4.925	4.8	**1.475**
Friedman test	Friedman's chi-squared = 50.918, df = 9, p-value = $7.235 \cdot 10^{-8}$									
APV Finner	0.00020	$< 10^{-5}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-5}$	$< 10^{-7}$	0.00035	0.00051	control

as one could expect, is the one which learns the optimal number of nearest neighbours between 1 and 100 (analogously as in RIONA and RIONIDA; see Tables 5.4 and 5.8). It should be noted that for the RIONA algorithm, we only tuned type of filter and the other parameters were fixed (the same as in RIONIDA). Keeping this in mind, three algorithms: BRACID, RIONA and kNN, have similar average ranks. The RIONA algorithm is significantly worse than the RIONIDA algorithm, but its performance is similar to BRACID and kNN.

From the above considerations, one can conclude that most likely (we used a reasonable variety of parameters but not all possibilities) selecting other than default parameters of algorithm and type of preprocessing filter (e.g. by tuning them to the used data sets) does not significantly change the situation presented for the *defG* strategy. In particular, it does not change the drawn conclusions that RIONIDA significantly outperforms all the algorithms used in comparisons. This is a quite astonishing result as it considerably strengthens the results presented for the *defG* strategy in favour of the RIONIDA algorithm.

Moreover, from the experiments performed for the considered *optG* strategy, one can observe what follows. The default parameters and type of filters (in *defG*) seem to be selected well (with qualities of algorithms comparable to those achieved in this strategy). In fact, some of the default settings (for other algorithms than RIONIDA) were used from this step. For example, MODLEM-C does not have default parameters, and we used as the default (in the previous step) the best-found parameters in this step. Also, the type of filter `SMOTE+ENN`, reported in [14] as the recommended selection (and confirmed in [152]), is also confirmed by this step. In fact, in Table 5.8, one can see that the optimal AF-learners generally contain this filter `SMOTE+ENN`.

Max strategy for G-mean (*maxG*)

Here, we will try to answer the following question: Could the general situation (with scores and ranks of algorithms) and conclusions, described in the previous subsubsections, change if parameters and type of filter were learned for algorithms (except RIONIDA) during the learning phase?

To answer this question, we assume that somehow for each learning algorithm, its optimal parameters and filters were selected separately for each data set taking also into account the fact that G-mean is the chosen performance measure. Here, 'optimal' relates to the *max* strategy described in Subsection 5.2.6 and the G-mean performance measure (such strategy was called earlier more specifically *maxG* strategy). Technically speaking, for each data set was chosen the maximal score out of all scores for different AF-learners (in the group). Let us recall that this strategy returns, in a sense, the upper bound of scores under the assumption that learning of parameters and filters was implemented for algorithms other than RIONIDA (and BRACID).

In Table 5.10, for each learning algorithm, the representative scores (for the *maxG* strategy) for all used data sets are given. Analogously as in the previous tables, $\text{RIONIDA}_{\text{G}}$ appears in the table instead of RIONIDA (which means that RIONIDA was set to optimise the G-mean measure). In the table, there is no information on AF-learners corresponding to the individual scores for different data sets. However, the readers were given a few such examples previously (see the description of the *max* strategy in Subsection 5.2.6). Certainly, all the scores in this table are higher

or equal to their corresponding scores in the *defG* and *optG* strategies (it follows straightforwardly from the formulation of the *max* strategy). For RIONIDA (and BRACID) the scores are the same as in the *defG* and *optG* strategies.

One can see from Table 5.10 that for this strategy still in most cases, the RIONIDA algorithm achieves the best score: for 20 data sets, 12 times it wins with all other algorithms. In situations when it loses with an algorithm, the difference between the best score and the score of RIONIDA is: once about 5%, once about 2%, once about 1%, three times below 0.5%, and twice below 0.1%. For these cases, RIONIDA has the following ranks: 4 (for *abalone*, *ionosphere*, *vehicle*), 2.5 (for *new-thyroid*), 2 (for *breast-w*, *glass*, *hepatitis*, *nursery*). These results are a little worse than those reported for the *defG* and *optG* strategies.

However, the results mentioned above again lead to the best outcome of the average rank for RIONIDA (1.725). The difference between the average rank of the RIONIDA algorithm and the second-lowest average rank (3.90 for kNN) is still (compared to the result for the previous strategy) relatively high (2.175). It should be noted that the worse average rank of BRACID in comparison to the previous strategy relates to the mentioned fact that for BRACID, one default AF-learner was used. Analogously, the average rank of RIONIDA is worse than in the previous strategy.

One can see from Table 5.10 that the Friedman statistical test gave again the p-value less than 10^{-7}. Since it is (much) smaller than 0.05, this means that there are significant statistical differences among compared algorithms. Therefore, we can proceed with a post-hoc test.

One can see in the discussed table that all the adjusted p-values are less than 0.05. It means that for each of the compared algorithms, even if it were possible to construct a meta-learning algorithm which for each data set would select the optimal AF-learner, it would be statistically worse than RIONIDA_G. It is worthy of underlining that this seems to be an impressive result.

One should observe, however, that for the group of kNN classifiers, the p-value is close to the threshold of 0.05. This means that the outperforming of kNN by RIONIDA is not very strongly supported by the Finner statistical test. On the other hand, one should keep in mind that kNN has a particular advantage over RIONIDA (and other used algorithms) due to the use of many (33) AF-learners.

5.3.2 Comparison of algorithms for F-measure

This subsection is analogous to Subsection 5.3.1 using F-measure instead of G-mean. The readers are referred to the previous section for details. Here, we only present a summary of the experimental setup:

- The performance measure we are interested in is F-measure.
- Thus, RIONIDA is set to optimise F-measure (RIONIDA_F is used).

Analogously as in Subsection 5.3.1, we present the results of comparative experiments for three strategies with F-measure as the performance measure: *defF*, *optF*, *maxF*.

Table 5.10: The values of G-mean (in %) for different algorithms and different data sets, for the *maxG* strategy (additionally for RIONIDA ranks are shown in parentheses). The RIONIDA algorithm was set to optimise the G-mean measure (i.e. $\text{RIONIDA}_{\text{G}}$ was used). It should be noted that for both $\text{RIONIDA}_{\text{G}}$ and BRACID, one default AF-learner was used (i.e. no filter and default parameters of the algorithm were used). For the other learning algorithms, the vector of representative scores was generated using the *maxG* strategy. For each data set, the best-obtained score is shown in bold. At the bottom are shown: (i) average rank for each algorithm, (ii) important outcomes of the Friedman statistical test (Friedman statistic, degrees of freedom, and p-value), and (iii) adjusted p-values (APV) with the Finner post-hoc test using RIONIDA as the control algorithm.

	The vectors of representative scores for different learning algorithms generated with the *maxG* strategy (for $\text{RIONIDA}_{\text{G}}$ and BRACID, one default AF-learner was used – their scores are the same as in the *defG* strategy)									
Data set	**kNN**	**PART**	**J48**	**RIPPER**	**RISE**	**MODLEM**	**MODLEM-C**	**RIONA**	**BRACID**	**$\text{RIONIDA}_{\text{G}}$**
abalone	63.46	71.78	70.85	**73.05**	60.03	65.54	55.06	59.91	65.80	67.94 (4)
balance-scale	65.34	56.74	23.37	15.45	40.97	12.90	2.78	33.26	58.68	**76.98 (1)**
breast-cancer	59.22	55.50	55.35	57.94	58.49	58.87	58.34	60.97	58.17	**64.98 (1)**
breast-w	97.46	96.35	95.79	96.21	97.02	96.16	94.89	**97.81**	96.91	97.53 (2)
car	86.23	95.45	90.23	80.81	76.90	89.09	89.24	85.29	87.47	**96.74 (1)**
cleveland	75.46	67.19	66.49	69.68	59.43	63.34	35.61	65.27	62.89	**76.38 (1)**
credit-g	65.66	66.78	66.17	65.59	65.27	65.65	66.87	66.35	62.27	**69.90 (1)**
ecoli	88.35	86.44	84.45	86.44	85.59	84.15	67.65	86.68	84.42	**88.82 (1)**
glass	68.03	66.66	**69.44**	57.99	54.69	63.82	47.64	67.69	39.90	69.26 (2)
haberman	59.88	61.75	63.80	62.48	60.35	62.64	57.10	61.00	59.55	**65.40 (1)**
hepatitis	**80.09**	69.80	70.02	68.07	71.16	73.42	68.06	73.46	77.11	79.00 (2)
ionosphere	92.41	89.01	87.97	88.28	**92.82**	88.72	89.60	90.62	91.42	90.89 (4)
mammography	88.28	85.04	83.41	84.51	84.57	79.96	70.60	74.17	85.41	**89.70 (1)**
new-thyroid	**98.94**	95.92	95.15	96.03	97.73	95.33	92.98	98.93	98.69	98.93 (2.5)
nursery	91.52	**99.96**	95.13	85.74	95.59	99.80	99.80	99.56	96.58	99.90 (2)
pima	71.48	69.26	70.34	70.03	68.70	69.45	70.83	67.13	71.28	**72.87 (1)**
postoperative	39.48	37.20	37.87	34.24	36.84	34.75	40.21	34.02	42.49	**43.66 (1)**
transfusion	62.90	62.62	64.93	64.38	63.11	62.31	58.84	63.34	64.39	**67.64 (1)**
vehicle	94.21	94.05	93.14	93.94	92.59	95.24	**95.55**	95.18	93.82	95.10 (4)
yeast	79.83	72.76	70.46	73.62	68.78	64.55	46.95	75.92	72.38	**84.95 (1)**
average rank	3.9	5.225	6.05	6.475	6.75	6.575	7.425	5.275	5.6	**1.725**
Friedman test	Friedman's chi-squared = 53.725, df = 9, p-value = $2.13 \cdot 10^{-8}$									
APV Finner	0.02310	0.00029	0.00001	$< 10^{-5}$	$< 10^{-6}$	$< 10^{-5}$	$< 10^{-7}$	0.00027	0.00008	control

Def strategy for F-measure (*defF*)

In this step, we again compare algorithms using their default AF-learners (as described in Subsection 5.2.6 for the *def* strategy), but this time with F-measure as the performance measure. Specifically, we compare the same AF-learners pointed out in the *defG* strategy in the previous section with one exception, namely $\langle \text{RIONIDA}_{\text{F}}, \texttt{Null-filter} \rangle$ instead of $\langle \text{RIONIDA}_{\text{G}}, \texttt{Null-filter} \rangle$.

In Table 5.12, for each learning algorithm, the representative scores of F-measure (for *defF*) for all used data sets are given.

One can see from this table that for half of the 20 data sets, RIONIDA wins with all other algorithms. In situations when it loses with an algorithm, the difference between the best score and the score of RIONIDA is: twice between 8%-10%, once about 4%, twice about 1%, and six times below 1% (including 4 times below 0.5%). For these cases, RIONIDA has the following ranks: 7 (for *abalone*, *glass*), 5 (for *ionosphere*), and 2 (for *breast-w*, *car*, *haberman*, *hepatitis*, *new-thyroid*, *nursery*, *yeast*).

The mentioned results are not as excellent as for the *defG* strategy, but still are very good in comparison to the other algorithms. In particular, RIONIDA again achieved the best average rank (2.15). It is smaller by 2.05 from the second-lowest average rank (4.2 for BRACID).

The Friedman statistical test returns again a very small p-value, i.e. (much) less than 0.05. This means that there exist statistical differences among compared algorithms; hence, one could perform the post-hoc Finner test for comparisons with the control algorithm RIONIDA.

All the adjusted p-values of the Finner procedure are less than 0.05. Thus, we can claim that RIONIDA is significantly better than any other algorithm used in the comparison. However, the highest p-value (around 0.03 for RIONIDA) is close to the threshold of 0.05. It shows that RIONIDA outperforms BRACID not as evidently as in the case for G-mean. Probably this is because BRACID was implemented to optimise F-measure. The second-highest p-value is for the RIONA algorithm (with its optimal AF-learner) and is smaller than 10^{-2}. All other p-values (related to other algorithms) are smaller than 10^{-3}.

Opt strategy for F-measure (*optF*)

In Table 5.11, the optimal (i.e. with the lowest average rank) AF-learners for the *optF* strategy are presented. If we compare these selected optimal AF-learners with the default AF-learners (used in the *defF* strategy presented above) the differences in settings (options or filters) can be observed for the following algorithms: kNN, PART, J48, RIPPER, MODLEM, MODLEM-C. For the others, the used AF-learners are the same in both strategies (*defF* and *optF*). Let us also note that some of the selected AF-learners are the same for both *optF* and *optG* strategies (for kNN, PART, RISE, MODLEM, RIONA, and naturally BRACID).

In Table 5.13, for each learning algorithm, the representative scores (for the *optF* strategy) for all used data sets are given. One can see from this table that for half of 20 data sets, RIONIDA wins with all algorithms (as in the *defF* strategy). In situations when it loses with an algorithm, the difference between the best score and the score of RIONIDA is: once above 17%, once above 8%, once about 5%, once above 2%, once above 1%, and five times below 1% (including 3 times below 0.5%). For these cases, RIONIDA has the following ranks: 8 (for *glass*), 7 (for *abalone*),

Table 5.11: The three best AF-learners (with the lowest average ranks shown in parentheses) selected by the *optF* strategy (for each algorithm). The best AF-learners are shown in bold and are used in the final comparison (for the *optF* strategy).

algorithm	three best AF-learners in each group
kNN	⟨**kNN** `-K 100 -X`, `SMOTE+ENN`⟩ **(35.8)**, ⟨kNN `-K 9`, `SMOTE`⟩ (37.1), ⟨kNN `-K 3`, `SMOTE+ENN`⟩ (38.2)
PART	⟨**PART -U**, `SMOTE+ENN`⟩ **(40)**, ⟨PART -U, `SMOTE`⟩ (41.275), ⟨PART `-C 0.5`, `SMOTE+ENN`⟩ (41.6)
J48	⟨**J48** `-C 0.1`, `SMOTE`⟩ **(39.525)**, ⟨J48 -A, `SMOTE`⟩ (41.275), ⟨J48 `-C 0.25`, `SMOTE`⟩ (41.275)
RIPPER	⟨**RIPPER** `-E -P`, `SMOTE+ENN`⟩ **(41.95)**, ⟨RIPPER -P, `SMOTE+ENN`⟩ (41.95), ⟨RIPPER , `SMOTE+ENN`⟩ (44.15)
RISE	⟨**RISE** , `SMOTE+ENN`⟩ **(38.475)**, ⟨RISE , `SMOTE`⟩ (42.2), ⟨RISE , `Null-filter`⟩ (60.325)
MODLEM	⟨**MODLEM** , `SMOTE`⟩ **(37.8)**, ⟨MODLEM , `SMOTE+ENN`⟩ (41.225), ⟨MODLEM , `Null-filter`⟩ (67.65)
MODLEM-C	⟨**MODLEM** `-M 6`, `Null-filter`⟩ **(52.575)**, ⟨MODLEM `-M 7`, `Null-filter`⟩ (52.675), ⟨MODLEM `-M 10`, `Null-filter`⟩ (52.925)
RIONA	⟨**RIONA** , `SMOTE+ENN`⟩ **(36.15)**, ⟨RIONA , `SMOTE`⟩ (40.5), ⟨RIONA , `Null-filter`⟩ (60.75)
BRACID	⟨**BRACID** , `Null-filter`⟩ **(26.8)**
RIONIDA	⟨**RIONIDA$_{\mathbf{F}}$** , `Null-filter`⟩ **(12.225)**

Table 5.12: The values of F-measure (in %) for different algorithms and different data sets, for *defF*. The RIONIDA algorithm was set to optimise F-measure (i.e. RIONIDA_F was used). For each data set, the best-obtained score is shown in bold. Also, for illustration, for five algorithms on the right (including RIONIDA_F), ranks for these algorithms and different data sets are shown (in parentheses). At the bottom are shown: (i) average rank for each algorithm, (ii) important outcomes of the Friedman statistical test (Friedman statistic, degrees of freedom, and p-value), and (iii) adjusted p-values (APV) with the Finner post-hoc test using RIONIDA as the control algorithm.

	The vectors of representative scores for different learning algorithms generated with the *defF* strategy									
Data set	**kNN**	**PART**	**J48**	**RIPPER**	**RISE**	**MODLEM**	**MODLEM-C**	**RIONA**	**BRACID**	**RIONIDA_F**
abalone	25.74	38.05	39.75	**42.35**	30.02	41.16 (2)	37.66 (5)	26.58 (9)	37.37 (6)	32.35 (7)
balance-scale	19.10	14.86	4.04	3.78	13.82	2.77 (9)	0.49 (10)	9.17 (6)	18.35 (3)	**34.30 (1)**
breast-cancer	44.78	39.80	40.78	39.41	44.31	42.90 (7)	44.79 (3)	43.05 (6)	45.52 (2)	**52.17 (1)**
breast-w	95.45	94.23	93.63	94.01	94.85	93.65 (8)	92.65 (10)	**96.39 (1)**	94.84 (5)	96.02 (2)
car	43.65	61.77	59.60	53.39	52.59	59.34 (6)	**85.88 (1)**	52.06 (9)	73.19 (3)	81.60 (2)
cleveland	33.33	34.55	36.04	37.75	31.55	35.35 (4)	16.80 (10)	35.01 (5)	33.36 (7)	**44.31 (1)**
credit-g	54.98	54.27	54.35	53.36	53.48	54.61 (5)	54.62 (3.5)	54.62 (3.5)	53.45 (9)	**58.27 (1)**
ecoli	59.63	57.66	56.93	60.05	59.52	59.28 (6)	53.05 (10)	58.40 (7)	59.87 (3)	**68.36 (1)**
glass	29.97	34.72	37.82	27.26	27.91	**38.28 (1)**	31.70 (5)	32.72 (4)	19.72 (10)	29.69 (7)
haberman	46.28	48.61	48.91	48.35	47.84	**50.06 (1)**	40.52 (10)	46.42 (7)	45.61 (9)	49.70 (2)
hepatitis	**60.64**	50.05	48.99	48.98	55.36	52.62 (6)	46.27 (10)	57.31 (4)	59.38 (3)	60.31 (2)
ionosphere	87.85	82.39	80.86	79.17	**88.71**	80.99 (8)	86.04 (6)	88.68 (2)	87.52 (4)	87.38 (5)
mammography	10.43	10.17	9.97	10.39	10.39	9.77 (9)	9.25 (10)	10.63 (3)	64.57 (2)	**67.33 (1)**
new-thyroid	94.82	90.77	91.49	90.94	92.44	89.85 (9)	88.12 (10)	95.36 (3)	**96.91 (1)**	96.40 (2)
nursery	53.08	91.54	74.03	68.46	75.36	95.40 (3)	**99.73 (1)**	75.74 (6)	95.10 (4)	98.92 (2)
pima	63.03	63.43	63.28	64.26	63.69	63.01 (8)	62.24 (10)	62.39 (9)	65.82 (2)	**66.04 (1)**
postoperative	24.13	19.73	20.99	17.49	20.10	18.59 (8)	23.55 (4)	17.65 (9)	31.93 (2)	**33.61 (1)**
transfusion	45.78	45.39	45.92	46.84	46.15	45.33 (9)	39.12 (10)	46.06 (5)	47.08 (2)	**50.02 (1)**
vehicle	84.44	86.08	85.14	85.96	83.55	84.44 (8.5)	89.74 (2)	86.10 (3)	85.81 (6)	**89.79 (1)**
yeast	37.77	33.98	35.40	37.37	40.03	37.32 (7)	29.00 (10)	38.14 (4)	**41.62 (1)**	41.24 (2)
average rank	5.475	6.1	6.3	6.425	5.825	6.225	7.025	5.275	4.2	2.15
Friedman test	Friedman's chi-squared = 38.785, df = 9, p-value = $1.26 \cdot 10^{-5}$									
APV Finner	0.00066	0.00007	0.00004	0.00004	0.00019	0.00005	$< 10^{-5}$	0.00124	0.03226	control

5 (for *ionosphere*), 3 (for *car*, *nursery*), and 2 (for *breast-w*, *hepatitis*, *new-thyroid*, *vehicle*, *yeast*). Thus, these results are a little bit worse than for the *defF* strategy presented above. In particular, ranks for 4 data sets are worse, and for 1 – better.

Regardless of this fact, RIONIDA again achieved the best average rank (2.3). It is smaller by 2.1 from the second-lowest average rank (4.4 for BRACID).

The Friedman statistical test again shows that there exist statistical differences among compared algorithms (with p-value much less than 0.05). This enables us to perform the post-hoc Finner test for comparisons with RIONIDA as the control algorithm.

Again, all the adjusted p-values of the Finner procedure are less than 0.05. Thus, we can again claim that RIONIDA is significantly better than any other algorithm used in the comparison (for the *optF* strategy). Again, the highest p-value (around 0.03 for BRACID) is relatively close to the threshold of 0.05. The second-highest p-value is for the kNN algorithm (with its optimal AF-learner), and it is smaller than 10^{-3}.

To sum up, for the *optF* strategy, RIONIDA achieves a little bit worse scores (in relation to other algorithms) and ranks than for the *defF* strategy. However, the statistical conclusion remains unchanged: RIONIDA is significantly better than any other compared algorithm (for the *optF* strategy).

Max strategy for F-measure (*maxF*)

In Table 5.14, for each learning algorithm, the representative scores (for the *maxF* strategy) for all used data sets are given. One can see from this table that for this strategy, the RIONIDA algorithm wins with other algorithms 6 times. This is not as good result as for the previous strategies (*defF* and *optF*, for which RIONIDA wins 10 times). However, one can observe that no other algorithm achieves such a result. The second best algorithms in this respect are kNN and MODLEM, which win with all other algorithms 3 times. Let us also discuss, as earlier, the situations when RIONIDA loses with an algorithm. In these cases, the difference between the best score and the score of RIONIDA is: once above 17%, once above 11%, once above 10%, 3 times between 3%-5%, and 8 times below 1% (including 5 times below or equal to 0.5%). For these cases, RIONIDA has the following ranks: 9 (for *glass*), 8 (for *abalone*), 5 (for *ionosphere* and *nursery*), 4 (for *car*), 3 (for *vehicle* and *yeast*), 2 (for *breast-w*, *ecoli*, *haberman*, *hepatitis*, *mammography*, *new-thyroid*, and *postoperative*). Thus, these results are undoubtedly worse than those for the *optF* strategy (and naturally worse than for *defF*). In particular, ranks for 10 data sets are worse (in one of these cases, the difference in comparison to *optF* is 2, and in the other cases, the difference is 1).

Regardless of this fact, RIONIDA again achieved the best average rank (2.85). It is smaller by 1.6 from the second-lowest average rank (4.45 for kNN).

The Friedman statistical test again shows the statistical differences among compared algorithms (with p-value much less than 0.05). This enables us to perform the post-hoc Finner test for comparisons with RIONIDA as the control algorithm.

However, the results related to all the previous cases (*defG*, *optG*, *maxG*, *defF*, and *optF*) concerning the p-values of the post-hoc Finner test are not repeated here. In one case, namely for kNN, the p-value is higher than 0.09. Thus, we cannot claim that RIONIDA is significantly better than kNN (for *maxF* strategy). On the other hand, let us also recall the previous discussion in the two last subsubsections

Table 5.13: The values of F-measure (in %) for different algorithms and different data sets, for the *optF* strategy (additionally for RIONIDA ranks are shown in parentheses). The RIONIDA algorithm was set to optimise F-measure (i.e. $\text{RIONIDA}_{\text{F}}$ was used). It should be noted that for both $\text{RIONIDA}_{\text{F}}$ and BRACID, one default AF-learner was used. For the other learning algorithms, the optimal AF-learner was selected using the *optF* strategy (and then it was used the same for all data sets). For each data set, the best-obtained score is shown in bold. At the bottom are shown: (i) average rank for each algorithm, (ii) important outcomes of the Friedman statistical test (Friedman statistic, degrees of freedom, and p-value), and (iii) adjusted p-values (APV) with the Finner post-hoc test using RIONIDA as the control algorithm.

	The vectors of representative scores for different learning algorithms generated with the *optF* strategy (for $\text{RIONIDA}_{\text{F}}$ and BRACID, one default AF-learner was used – their scores are the same as in the *defF* strategy)									
Data set	**kNN**	**PART**	**J48**	**RIPPER**	**RISE**	**MODLEM**	**MODLEM-C**	**RIONA**	**BRACID**	**$\text{RIONIDA}_{\text{F}}$**
abalone	25.74	37.82	38.90	**40.79**	30.02	39.34	37.51	26.58	37.37	32.35 (7)
balance-scale	11.25	19.72	2.17	0.28	13.82	0.00	0.50	9.17	18.35	**34.30 (1)**
breast-cancer	44.78	41.30	41.57	43.44	44.31	45.24	43.74	43.05	45.52	**52.17 (1)**
breast-w	95.45	94.32	93.35	94.18	94.85	93.80	92.53	**96.39**	94.84	96.02 (2)
car	49.61	60.24	68.10	53.34	52.59	84.07	**86.66**	52.06	73.19	81.60 (3)
cleveland	33.44	38.57	26.09	38.13	31.55	23.20	15.71	35.01	33.36	**44.31 (1)**
credit-g	54.98	54.72	53.01	53.67	53.48	53.52	54.64	54.62	53.45	**58.27 (1)**
ecoli	59.22	58.01	62.76	60.65	59.52	61.64	53.05	58.40	59.87	**68.36 (1)**
glass	31.98	31.74	42.30	29.70	27.91	**47.21**	31.65	32.72	19.72	29.69 (8)
haberman	46.40	48.49	48.52	48.81	47.84	40.93	39.58	46.42	45.61	**49.70 (1)**
hepatitis	**61.58**	52.45	46.48	49.99	55.36	53.27	46.82	57.31	59.38	60.31 (2)
ionosphere	**90.03**	82.64	80.66	77.16	88.71	80.57	86.24	88.68	87.52	87.38 (5)
mammography	10.42	10.27	63.21	10.00	10.39	67.30	7.84	10.63	64.57	**67.33 (1)**
new-thyroid	94.97	91.20	90.82	90.05	92.44	92.30	89.57	95.36	**96.91**	96.40 (2)
nursery	55.94	92.56	72.88	72.76	75.36	99.42	**99.73**	75.74	95.10	98.92 (3)
pima	63.03	63.16	62.97	63.97	63.69	63.34	62.36	62.39	65.82	**66.04 (1)**
postoperative	24.13	16.65	19.96	18.02	20.10	17.43	20.67	17.65	31.93	**33.61 (1)**
transfusion	45.78	45.36	47.07	45.89	46.15	40.37	39.55	46.06	47.08	**50.02 (1)**
vehicle	84.44	86.72	86.42	86.85	83.55	89.57	**90.07**	86.10	85.81	89.79 (2)
yeast	38.72	34.62	37.15	39.87	40.03	34.21	28.59	38.14	**41.62**	41.24 (2)
average rank	5.5	6.1	6.3	6.1	5.8	5.7	7.05	5.75	4.4	**2.3**
Friedman test	Friedman's chi-squared = 33.611, df = 9, p-value = 0.0001045									
APV Finner	0.00093	0.00022	0.00013	0.00022	0.00046	0.00049	0.00001	0.00047	0.02828	control

of Subsection 5.2.6 (page 162). Thus, it is reasonable to expect that a potential meta-learning algorithm for kNN (selecting for each data set the optimal AF-learner), would achieve worse results than presented in Table 5.14. For the other compared algorithms the p-values are less than 0.05 (for PART slightly above 0.01, and for all other algorithms below 0.01).

To sum up, for the *maxF* strategy, RIONIDA achieves noticeably worse scores (in relation to other algorithms) and ranks than for the *optF* strategy. However, the statistical conclusion concerning comparisons with other algorithms remains nearly unchanged: RIONIDA is significantly better than each of compared algorithms, excluding kNN (for the *maxF* strategy).

Table 5.14: The values of F-measure (in %) for different algorithms and different data sets, for the *maxF* strategy (additionally for RIONIDA ranks are shown in parentheses). The RIONIDA algorithm was set to optimise F-measure (i.e. $\text{RIONIDA}_{\text{F}}$ was used). It should be noted that for both $\text{RIONIDA}_{\text{F}}$ and BRACID, one default AF-learner was used (i.e. no filter and default parameters of the algorithm were used). For the other learning algorithms, the vector of representative scores was generated using the *maxF* strategy. For each data set, the best-obtained score is shown in bold. At the bottom are shown: (i) average rank for each algorithm, (ii) important outcomes of the Friedman statistical test (Friedman statistic, degrees of freedom, and p-value), and (iii) adjusted p-values (APV) with the Finner post-hoc test using RIONIDA as the control algorithm.

	The vectors of representative scores for different learning algorithms generated with the *maxF* strategy (for $\text{RIONIDA}_{\text{F}}$ and BRACID, one default AF-learner was used – their scores are the same as in the *defF* strategy)									
Data set	**kNN**	**PART**	**J48**	**RIPPER**	**RISE**	**MODLEM**	**MODLEM-C**	**RIONA**	**BRACID**	**$\text{RIONIDA}_{\text{F}}$**
abalone	26.85	38.88	40.17	**43.40**	32.58	41.16	37.86	26.58	37.37	32.35 (8)
balance-scale	23.91	19.81	4.22	4.40	13.82	2.77	0.50	9.17	18.35	**34.30 (1)**
breast-cancer	47.95	42.10	42.00	45.32	44.81	45.24	44.79	47.49	45.52	**52.17 (1)**
breast-w	95.75	94.32	93.69	94.18	95.39	93.80	92.65	**96.39**	94.84	96.02 (2)
car	50.49	**91.76**	76.43	67.12	68.37	88.34	88.34	77.18	73.19	81.60 (4)
cleveland	40.21	38.57	36.10	38.94	31.55	35.35	16.80	35.01	33.36	**44.31 (1)**
credit-g	54.98	54.72	54.35	53.68	53.48	54.61	54.64	54.62	53.45	**58.27 (1)**
ecoli	**69.20**	60.65	62.76	64.70	61.54	61.64	53.13	62.20	59.87	68.36 (2)
glass	32.87	40.55	42.86	33.62	29.86	**47.21**	31.89	35.68	19.72	29.69 (9)
haberman	46.72	48.61	49.00	48.81	47.84	**50.06**	40.52	46.42	45.61	49.70 (2)
hepatitis	**64.27**	53.86	52.80	49.99	55.36	53.27	46.90	57.31	59.38	60.31 (2)
ionosphere	90.65	86.76	85.32	85.81	**91.12**	87.08	87.08	89.19	87.52	87.38 (5)
mammography	65.52	60.84	63.30	65.02	**67.83**	67.30	62.26	60.84	64.57	67.33 (2)
new-thyroid	95.33	92.92	91.61	91.87	95.56	92.30	89.57	95.85	**96.91**	96.40 (2)
nursery	57.43	99.62	87.59	73.34	95.28	**99.73**	**99.73**	98.98	95.10	98.92 (5)
pima	63.43	63.48	63.34	64.26	63.69	63.34	62.74	62.39	65.82	**66.04 (1)**
postoperative	**38.19**	21.70	21.67	18.02	20.10	18.59	23.55	17.65	31.93	33.61 (2)
transfusion	46.10	48.95	47.67	46.84	46.15	45.33	40.18	46.06	47.08	**50.02 (1)**
vehicle	88.15	89.77	87.55	88.98	86.55	89.70	**90.22**	90.16	85.81	89.79 (3)
yeast	41.60	34.77	37.24	39.98	40.29	37.32	29.00	39.57	**41.62**	41.24 (3)
average rank	4.45	5.175	6.575	6	5.85	5.45	7.325	5.675	5.65	**2.85**
Friedman test	Friedman's chi-squared = 28.68, df = 9, p-value = 0.0007337									
APV Finner	0.09469	0.01705	0.00045	0.00300	0.00388	0.00850	0.00003	0.00570	0.00570	control

5.3.3 Conclusions for G-mean and F-measure

To sum up, we performed experiments to compare our new proposed RIONIDA algorithm with the selected nine state-of-the-art algorithms with possible use of two state-of-the-art filters. For each algorithm, we used a reasonable number of AF-learners (variations of algorithms options and preprocessing filters). For two algorithms, RIONIDA and BRACID, we used only one AF-learner (i.e. these algorithms were always used with their default options and without filter).

We performed comparative experiments using two performance measures: G-mean and F-measure. For each of these measures, we compared RIONIDA with other algorithms using three strategies: *def*, *opt* and *max*. On the one hand, the *def* strategy is more appropriate for the algorithms with no variability in AF-learners, in particular for BRACID. On the other hand, the strategies *opt* and *max* give an advantage to the algorithms with more than one AF-learner.

For G-mean, it was shown that for any of the three strategies, RIONIDA significantly outperforms any algorithm used in the comparison. For F-measure, it was shown the same with one exception (for the *maxF* strategy and the kNN algorithm; in this case, RIONIDA achieved better average rank than kNN but we cannot claim that the difference between RIONIDA and kNN is statistically significant).

It means that regardless of whether we use default settings of algorithms or adjusted settings or even (potentially) learned settings by the meta-learning scheme, RIONIDA outperforms significantly any of these algorithms (with one mentioned exception) for the chosen set of real-life data sets. These obtained experimental results seem to be exceptionally good. Taking into account the thorough preparation of experiments (see Subsections 5.1.1-5.1.5 and Subsections 2.6.1-2.6.5), the presented results indicate that the newly presented RIONIDA algorithm most likely will be also competitive with these algorithms for other real-life classification tasks with imbalanced data.

From the performed experiments, it also follows that the RIONIDA algorithm can adapt to different performance measures (at least the two of them) very well. Thus, we may suppose that the presented results would extend for other performance measures based on the confusion matrix. This would make the RIONIDA algorithm very universal with a possibility e.g. to embed into it any such performance measure defined by a user for a particular classification task.

5.4 Additional comments on experiments

In this section, we present some additional comments on the performed experiments, which can help to understand why the RIONIDA algorithm outperforms some well-known methods dealing with imbalanced data. At the same time, we explain some advantages of RIONIDA. The included comments are presented to give the readers better intuition about RIONIDA performance and its quality rather than all details. This is done to make the presentation more compact. Finally, we analyse the real running time of RIONIDA, which was measured during the experiments presented in the previous section. In particular, we present a comparison of it with

other algorithms used in the experiments.

5.4.1 Studying the role of RIONIDA components

The key component of RIONIDA is the estimation of its performance for each possible triple of internal parameters $(k, p, s) \in K \times P \times S$ (see Section 4.4; also see Subsection 4.6.2). The analysis of the significance of all these parameters k, p, s was done after performing the comparative experiments. However, this analysis (using the whole available data sets) was presented earlier in Chapter 4 while presenting the RIONIDA algorithm. The significance of the parameters k, p, s was shown in Subsections 4.3.2, 4.3.3, 4.3.5, respectively. The estimation of the optimal values of these parameters is done very precisely in the learning phase since the validation process is performed by the leave-one-out method on the whole training set. Thus, in a sense, the full information for the given training set is used in the process of tuning these internal parameters. It is worth mentioning that the time complexity of this process is relatively low due to using the dynamic programming technique. All this means that the RIONIDA algorithm can learn the relevant values of its internal parameters very efficiently and precisely (and so proves to be highly effective). In our opinion, this is one of the main advantages of the RIONIDA algorithm.

5.4.2 The *balance-scale* data set and outliers

Out of the data sets used in the experiments, we would like to turn out the readers' attention to the *balance-scale* data set. For most of the objects from the minority class of this data set, the objects closest to them belong to the majority class. Such objects are called outliers (see Subsection 2.4.3). In other words, in the considered data set, most of the objects from the minority class are outliers. This is the reason why this data set is considered as a very hard imbalanced learning problem in [154].

Also, we performed separate experiments for RIONIDA with the fixed parameter $s = 1.0$ (which relates to the pure rule-based approach). In this case, for the minority class, generally, no (consistent) rules were found. The fact that most of the objects are outliers explains this fact. It also provides an intuition why algorithms which use standard rules can construct classifiers with poor quality for such data sets.

However, RIONIDA in most of the cross-validation splits (in a quite stable way – see next subsubsection) finds the optimal value $s = 0.5$. This corresponds to the situation that original rules may be inconsistent, but after changing the coverage region of the rule by half are becoming consistent. Generally, if we take into account objects from the minority class, the rules with decreasing value of the parameter s enable us to increase the Sensitivity of the rule.

It is an example illustrating that the RIONIDA algorithm can deal with data sets containing many outliers. This fact can be regarded as a powerful advantage of the RIONIDA algorithm.

5.4.3 Analysis of the optimal values of parameters obtained in the learning phase of RIONIDA

During the experiments presented in the previous section, we saved the internally learned optimal values of the parameters k, p, and s obtained during the learning phase in different runs of the performed experiments (10 times repeated 10-fold stratified cross-validation process) for RIONIDA. Thus, we obtained 100 triples of the optimal parameter values.

It can be informative to check for particular data sets whether the learned optimal parameters are 'stable' in different runs of RIONIDA. This can be relevant for at least three reasons.

First, stability (of one or more parameter) for a fixed domain may indicate that specific values of parameters are appropriate globally for all objects from that domain. This may mean that for future additional training (currently unknown) objects from that domain no further learning of (one or more) parameters is needed. Also, small fluctuations of (one or more) optimal values of parameters may indicate that for future training objects (currently unknown) from that domain the learning could be limited to a smaller range of some (one or more) parameters. Such limitation can influence the learning time and space allocations of the algorithm. This can be essential for scalability of the RIONIDA algorithm. For example, this can be crucial for the application of RIONIDA to so-called big data (see e.g. [61]). In the considered case of stability of parameters, learning of the optimal parameters could be done for a relatively small part of the data set (see Section 6.2).

Second, the stability of (one or more) parameter can be an argument for the quality of the obtained classifier. The more stable optimal value of the parameter is, the more reliable resulting classifier can be regarded as.

Third, in case of stability (of one or more parameters), the values of stable parameters may be a kind of description of a domain. For example, a domain can be described as 'more appropriate for rule-based methods' or 'more appropriate for instance-based methods'.

In Table 5.15, we present the averages and standard deviations of the optimal values of the parameters k, p, and s obtained in the mentioned experiments for RIONIDA (for both $\text{RIONIDA}_{\text{G}}$ and $\text{RIONIDA}_{\text{F}}$). In the current analysis, we focus on $\text{RIONIDA}_{\text{G}}$, and therefore RIONIDA will refer to this setting. In current considerations, the most interesting for us is the standard deviation. In this case, the small value of this measure means that in most (or all) runs of RIONIDA the learned optimal values of parameter were similar (or even equal).

Let us describe some conclusions for a few exemplary data sets and information from this table (and also from direct observations of the learned optimal values of parameters).

For *balance-scale* data set, the learned optimal parameters seem 'the most stable'. In all runs of RIONIDA, the learned optimal values of the parameters k and s were equal to 9 and 0.5, respectively. The learned optimal value of the parameter p was around value 0.1. It was equal to 0.08 (15 times), 0.09 (31), 0.1 (8), 0.11 (11), 0.12 (34), or 0.13 (1 time).

For *hepatitis*, *ionosphere*, *transfusion*, and *vehicle* data sets the learned optimal

Table 5.15: Table presenting fluctuations of optimal values of parameters k, p, s among different runs (different splits in the cross-validation schemes) of RIONIDA (RIONIDA$_{\mathrm{G}}$ and RIONIDA$_{\mathrm{F}}$) for each data set used in experiments. Averages and standard deviations of optimal parameters k, p, and s are rounded to integers, two decimals, and one decimal, respectively.

	RIONIDA$_{\mathbf{G}}$			**RIONIDA$_{\mathbf{F}}$**		
	Averages and standard deviations of optimal parameters					
Data set	k	p	s	k	p	s
abalone	76 ± 23	0.08 ± 0.01	0.9 ± 0.1	53 ± 21	0.15 ± 0.02	1.0 ± 0.1
balance-scale	9 ± 0	0.10 ± 0.02	0.5 ± 0.0	9 ± 1	0.13 ± 0.02	0.5 ± 0.0
breast-cancer	67 ± 22	0.28 ± 0.03	0.2 ± 0.3	67 ± 21	0.28 ± 0.03	0.2 ± 0.3
breast-w	16 ± 22	0.12 ± 0.09	-0.1 ± 0.1	17 ± 23	0.13 ± 0.09	-0.1 ± 0.1
car	33 ± 29	0.18 ± 0.14	0.4 ± 0.4	59 ± 37	0.24 ± 0.12	0.8 ± 0.4
cleveland	62 ± 22	0.12 ± 0.02	0.4 ± 0.6	81 ± 14	0.16 ± 0.02	0.8 ± 0.4
credit-g	62 ± 19	0.30 ± 0.02	0.6 ± 0.4	60 ± 19	0.30 ± 0.02	0.6 ± 0.4
ecoli	81 ± 28	0.26 ± 0.03	0.7 ± 0.3	27 ± 11	0.37 ± 0.03	0.7 ± 0.4
glass	12 ± 8	0.08 ± 0.05	0.2 ± 0.4	4 ± 5	0.07 ± 0.11	0.0 ± 0.2
haberman	81 ± 18	0.19 ± 0.03	0.6 ± 0.2	80 ± 16	0.18 ± 0.03	0.6 ± 0.2
hepatitis	35 ± 14	0.14 ± 0.03	-0.1 ± 0.0	31 ± 17	0.18 ± 0.06	-0.1 ± 0.1
ionosphere	7 ± 3	0.02 ± 0.03	-0.1 ± 0.0	7 ± 3	0.06 ± 0.06	-0.1 ± 0.0
mammography	73 ± 31	0.03 ± 0.01	0.8 ± 0.3	8 ± 3	0.26 ± 0.04	0.1 ± 0.4
new-thyroid	67 ± 22	0.11 ± 0.03	-0.1 ± 0.1	66 ± 22	0.11 ± 0.03	-0.1 ± 0.1
nursery	33 ± 33	0.23 ± 0.13	0.8 ± 0.5	27 ± 35	0.17 ± 0.17	0.5 ± 0.6
pima	44 ± 21	0.32 ± 0.03	0.8 ± 0.4	60 ± 20	0.29 ± 0.03	0.8 ± 0.4
postoperative	13 ± 12	0.19 ± 0.09	0.3 ± 0.5	17 ± 12	0.16 ± 0.08	0.0 ± 0.3
transfusion	28 ± 15	0.24 ± 0.02	-0.1 ± 0.0	31 ± 14	0.28 ± 0.04	-0.1 ± 0.0
vehicle	6 ± 3	0.21 ± 0.13	-0.1 ± 0.0	4 ± 3	0.28 ± 0.19	-0.1 ± 0.0
yeast	54 ± 19	0.03 ± 0.01	1.0 ± 0.1	37 ± 21	0.21 ± 0.03	0.6 ± 0.5

parameter s in all considered runs of RIONIDA was constant and equal to -0.1. As $s = -0.1$ corresponds to the pure instance-based method in RIONIDA (see Subsection 4.3.5), one can describe these data sets as 'more appropriate for instance-based methods'. For *new-thyroid* data set the value of the optimal parameter s can be considered as very stable (98 times it was equal to -0.1, once to 0.5, and once to 0.9). Also, for *breast-w* data set, the value of the optimal parameter s can be considered as very stable (99 times it was equal to -0.1, and once to 1.0). Hence, these two data sets can also be described as 'more appropriate for instance-based methods'.

On the other hand, *yeast* data set has a very stable value of the optimal parameter s around value 1.0. It was equal to 1.0 in 98 cases, 0.5 in one case, and 0.7 in one case. As $s = 1.0$ corresponds to the pure rule-based method in RIONIDA (see Subsection 4.3.5), this data set can be described as 'more appropriate for rule-based methods'. For *abalone* data set the value of the optimal parameter s can be considered as very stable around value 0.9. It was equal to 1 (81 times), 0.9 (5), 0.8 (3), 0.7 (7),

0.6 (1), 0.5 (1), 0.4 (1), or 0.3 (1 time). Also, this data set can be described as 'more appropriate for rule-based methods'.

As it was mentioned, for *balance-scale* data set, the value of the optimal parameter s was constant for all considered runs of RIONIDA and was equal to 0.5. It is an interesting example for the data set which can be described as 'data set appropriate for methods between instance- and rule-based methods'. Another example of such a case is *haberman* data set. Its optimal value of the parameter s was around value 0.6. It was equal to 0.6 (28 times), 0.5 (20), 0.7 (2), 0.4 (5), 0.8 (4), 0.3 (9), 0.9 (28), and 0.2 (4 times). See also Subsection 4.3.5 for considerations on *haberman* data set (and Figures 4.12, 4.13, 4.14) taking into account the whole available data and dependence of G-mean on the parameter s.

Now, let us take into account the fluctuations of the optimal values of the parameter p. As can be seen in Table 5.15, for most of the considered data sets, this value has a relatively small standard deviation (for *yeast*, *abalone*, *mammography*, *balance-scale*, *credit-g*, *cleveland*, *transfusion*, *breast-cancer*, *hepatitis*, *haberman*, *new-thyroid*, *ionosphere*, *pima*, and *ecoli* data sets). In these cases, the learned optimal value of the parameter p can be considered as stable.

What about the cases when the optimal values of parameters are unstable? These could be investigated in two directions.

First, one could check how fluctuations of the optimal values of parameter change the value of considered performance measure. This issue was somehow investigated globally (for the whole data sets) in Subsections 4.3.2, 4.3.3, 4.3.5. However, further investigation could be done (see Section 6.2).

Second, such fluctuations may suggest that searching not globally but locally for the optimal values of parameters (separately for different regions of a considered domain) could potentially increase the quality of RIONIDA performance (see Section 6.2).

Let us now concentrate on the average value of the optimal parameter p from Table 5.15. It should be noted that for many considered data sets (11 out of 20) this value is very close to the percentage of the minority class in the data set (see Table 5.1). (Moreover, for these 11 data sets except *breast-cancer* these differences are less than the standard deviation of the value of the optimal parameter p.) The distance between these two values is:

- less than 1% for *glass*, *yeast*, *credit-g*, *mammography*, *transfusion*, *abalone*, and *cleveland* data sets;
- less than 3% for *breast-cancer*, *balance-scale*, *pima*, and *vehicle* data sets.

The presented observations are consistent with Theorem 4.1, formulated and proved in the thesis. This fact could be used in future research for using the default candidate for the optimal value of the parameter p. This fact could also be used to search for the optimal value of this parameter around the default value (see Section 6.2). Such an approach could be especially useful for big data sets.

On the other hand, it should be noted that there exist a few data sets for which the difference between the average optimal value of the parameter p and the percentage of the minority class in data set is relatively high: *ionosphere* (approximately 34%

of difference), *breast-w* (22%), *nursery* (20%), *ecoli* (15%). This fact proves the usefulness of learning of the optimal value of the parameter p in general (see also Subsection 4.3.3).

5.4.4 Analysis of running time of RIONIDA

In Subsection 4.5.2, we calculated the time complexity of the learning phase of RIONIDA. Here, we check whether the time of learning phase in practice reaches the (pessimistic) theoretical time complexity. In the performed experiments, the number of conditional attributes was relatively small (from 4 to 34). Also, the sets P and S (i.e. sets of admissible values of the parameters p, s) are constant in our comparative experiment. Thus we omit these three factors in the analysis of the time of learning phase. In the performed experiments, the number of objects in data sets was varying from 90 to 12960 examples in total. We make the running time analysis only for this factor.

Figure 5.8 presents the relationship between the size of the data set and the learning phase duration of RIONIDA. Each data set is represented on this figure by point (x, y), where $x =$ number of examples in data set, $y =$ time of learning phase.

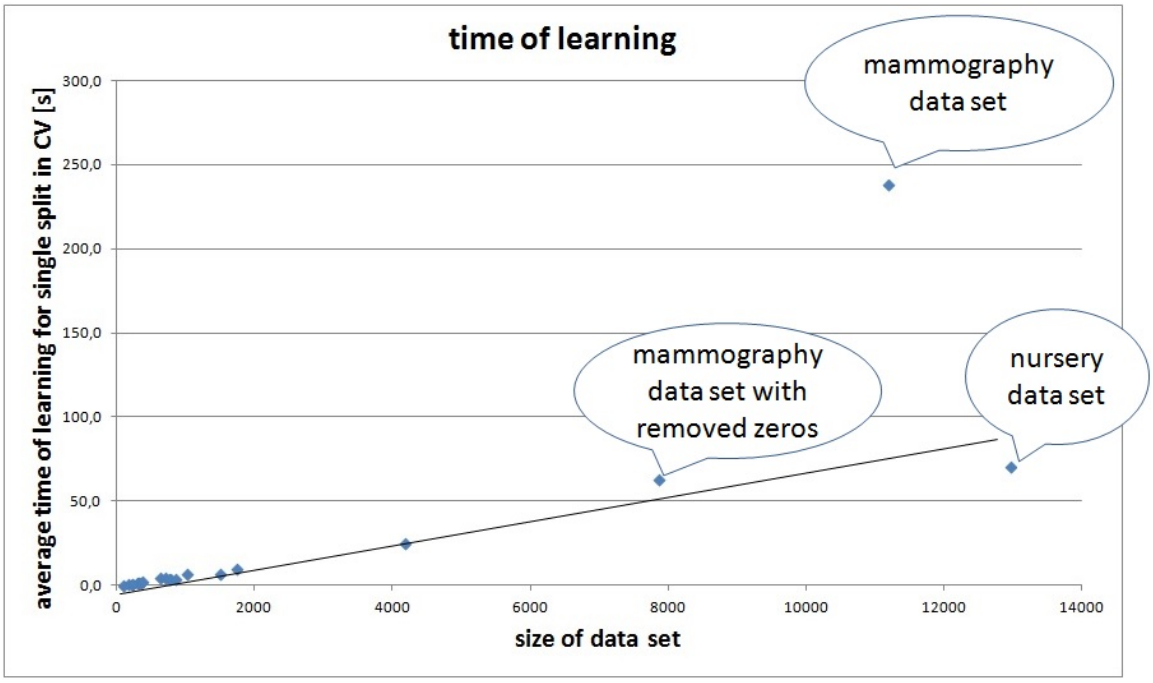


Figure 5.8: Dependence of the learning phase duration of RIONIDA (in seconds) on the size of data set (i.e. the number of objects in data set). The figure presents the average time of learning phase of RIONIDA for a single split in the 10-fold stratified cross-validation. In any split, the training set contains roughly 90% of the data set.

First, let us exclude from our considerations the *mammography* data set (we discuss this case below). In this context, it is visible, that the points roughly lay on the straight line. At first glance, it is a surprising observation since the (pessimistic) theoretical time complexity is a quadratic function of the number of training examples. Below we explain it and add further comments.

Let us recall that the time complexity of the learning phase is $O(mn^2 + n|S| \cdot k_{max} \cdot (mk_{max} + |P|))$, where $n = |trnSet|$, $m = |A|$, k_{max} is the parameter used to define the maximal size of the neighbourhood to be analysed ($k_{max} = |K|$), P, S are sets of admissible values of the parameters p, s, respectively (see Theorem 4.3 in Subsection 4.5.2). In our primary experiments $|P| \leq mk_{max}$ holds, and as a consequence, the time complexity is $O(m(n^2 + n|S| \cdot k_{max}^2))$. For the data sets used in our experiments $n < 13000$. Since in our primary experiments $|S| = 12$, $k_{max} = 100$, then $n^2 < n|S| \cdot k_{max}^2$. In other words, for the used data sets and settings, the factor $n|S| \cdot k_{max}^2$ is dominant over the factor n^2. This fact explains the observed in the performed experiments the 'linearity' of the time of learning phase relative to n. The quadratic factor will become dominant for $n > 120000$.

On the other hand, the quadratic time complexity relates to the searching for k_{max} nearest objects to the considered training example (or more objects in the specific situation described in Definition 2.14) among n objects (see Subsection 4.5.2). We assumed that this operation could be done in the linear time relative to n (see Subsections 4.5.2, 3.4.1). However, in our implementation, we use indexing trees for speeding up this operation (see Subsection 3.6.1). It was experimentally shown in [219, pp.77-78] that by using indexing trees: (1) this operation is faster than linear, (2) the acceleration (of this operation) significantly grows with growing n. Also, it was (experimentally) shown that the time of constructing indexing trees is significantly shorter than the time of (multiple) searching of nearest neighbours in a data set. All these facts were not analysed theoretically; thus, we can only say that in our implementation, instead of factor n^2, occurs a factor with time complexity between linear and quadratic. This fact appears promising in the context of a potential need for scalability of RIONIDA.

Let us return to the case of the *mammography* data set. It is visible in the presented figure that for this data set the learning phase is a few times higher than for the *nursery* data set with a larger size of data set. This is due to two reasons. First, the neighbourhood N may contain more objects than k (for the specific situation described in Definition 2.14). Second, *mammography* data set contains a large number of objects described with the same values of attributes (precisely 3329 objects described by zeros for all conditional attributes; most of them belong to the majority class and 7 of them to the minority class); consequently, the neighbourhood N contains much more objects than k (around 30% of neighbourhoods contain around $0.3 \cdot n$ objects). It seems that these objects are kind of artefacts and should be removed from the data set before analysis. However, we use this data set analogously as it was done in [37] and subsequent papers[10].

Only for the sake of this subsection, we performed an additional experiment for RIONIDA and *mammography* data set with all objects described by zeros in conditional attributes removed (it is also presented and described on the figure). The performed experiment shows that for such modified data set the time of learning phase significantly decreases and confirms the roughly linear time of learning phase relative to the size of the training set (at least for the considered data sets).

We also analysed the testing time of a single object for each data set. The average time of testing of a single object for different data sets was between 0.03ms and 0.35ms

[10] Also, prof. Nitesh Chawla, one of the co-authors of [37] suggested me using this data set as is.

(parts of milliseconds) and 15.16 ms for *mammography* data set.

First, let us again exclude from our considerations the *mammography* data set (we discuss this case below). We observed that the average time of testing of a single object for the larger data sets used in our experiments (*abalone* and *nursery*) is comparable to the case for the smaller data sets. Thus, we repeated the experiments without using indexing trees to check whether significant acceleration is achieved by using this specialised data structure.

Figure 5.9 shows[11] the average time of testing of a single object for (1) standard version of RIONIDA with use of indexing trees, and (2) version of RIONIDA without using indexing trees. In the case without using indexing trees, one can observe the roughly linear dependence of the average time of testing of a single object on the size of data sets. This observation is consistent with the theoretical time complexity of testing operation for RIONIDA (see Subsection 4.5.1). On the other hand, for the version with the use of indexing trees, one can observe variability between two constant values. The plot in this figure suggests dependence close to a constant value. However, we must admit that we used rather small data sets to draw any far-reaching conclusions about the (experimental) dependence of the average time of testing of a single object (for RIONIDA) on the size of data sets. Regardless, this shows that even for data sets used in our experiments, which are relatively small, the significant acceleration of the testing phase for RIONIDA by using indexing trees is achieved. This is a promising fact for attempting to analyse big data sets.

Taking into account the above considerations, let us come back shortly to the case of the time of learning phase. It should be noted that for the testing phase, the operation of searching for nearest objects is dominant. As it was mentioned, this (repeated) operation will become dominant for larger data sets also in the learning phase. If this operation for larger data sets would also take time close to constant, it would strongly affect the time complexity of the learning phase. Thus, the observations for the considered data sets justify the mentioned supposition that, for the learning phase, the practical time complexity can be close to linear. To be more precise, this issue needs further investigation in the future (see Section 6.2).

Let us return to the case of the *mammography* data set. For example, the testing phase for this data set takes around 210 times more than for the *nursery* data set with the larger size of data set. The reason for this fact is analogous to the one discussed above for the learning phase. Here, we see that the effect of large neighbourhoods can slow down the testing phase of RIONIDA far more.

As it was mentioned in Subsection 3.3.2, one could consider dedicated data structures for grouping objects with identical attribute values for speeding up searching for the neighbourhood N in such situations as described above for the *mammography* data set (see Section 6.2).

Finally, we compare the time of computations for different learning algorithms. As an example, we use AF-learners (combinations of algorithms and filters) presented in Table 5.8 (the optimal AF-learners for the *optG* strategy) on page 168. Thus, in

[11] This experiment was performed on another computer than the other experiments presented in the thesis. Thus the absolute values of times can differ with other presented data or experiments using time factor. However, our desire is to recognise the relative difference between the times for considered two cases.

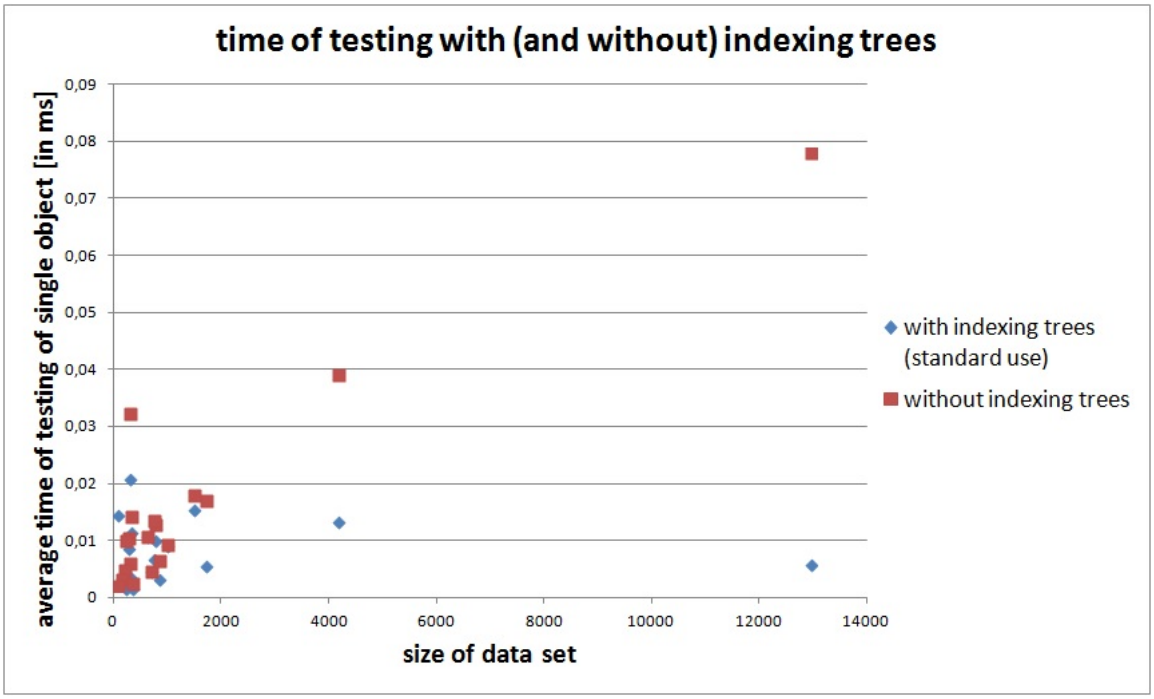


Figure 5.9: Dependence of the average time of testing of a single object (in milliseconds) on the size of data set (number of objects in data set) for: (1) standard version of RIONIDA with use of indexing trees, and (2) version of RIONIDA without using indexing trees.

particular, MODLEM, BRACID, RIONIDA are used without filtering; MODLEM-C – with the `SMOTE` filter; and the remaining learning algorithms – with `SMOTE+ENN`.

First, we present the time of learning phase (generally the most time-consuming phase). We computed the average training time for a single split in the 10-fold stratified cross-validation for each learning algorithm used in the comparative experiments. We summed these values for all data sets used in the experiments excluding *mammography* data set (due to described above repetitions of objects in it). During computations separately were computed: (1) time of preprocessing of data sets (using filters) and (2) pure learning time of the learning algorithm.

In Figure 5.10, we present the time of learning phase for each learning algorithm computed in above mentioned way. In this figure, we distinguish the mentioned preprocessing time and pure learning time. Learning algorithms which do not use filtering (MODLEM-C, BRACID, RIONIDA) have no visible time of filtering part of the learning phase. However, also the MODLEM algorithm has insignificant (in time) filtering part. This is due to the use of `SMOTE` filter, much faster than `SMOTE+ENN`. From this figure, we can conclude that on average training time of RIONIDA is comparable to other algorithms. Here, RIONIDA is in the second place, after MODLEM-C (which performs a few times faster)[12]. Moreover, it is worth recalling that the learning phase of the used in experiments implementation of RIONIDA can be accelerated a few times (see Subsection 4.5.3).

[12]In the case without excluding *mammography* data set, RIONIDA would be on the one before last place. However, as it was mentioned above it is possible to accelerate RIONIDA for such data sets.

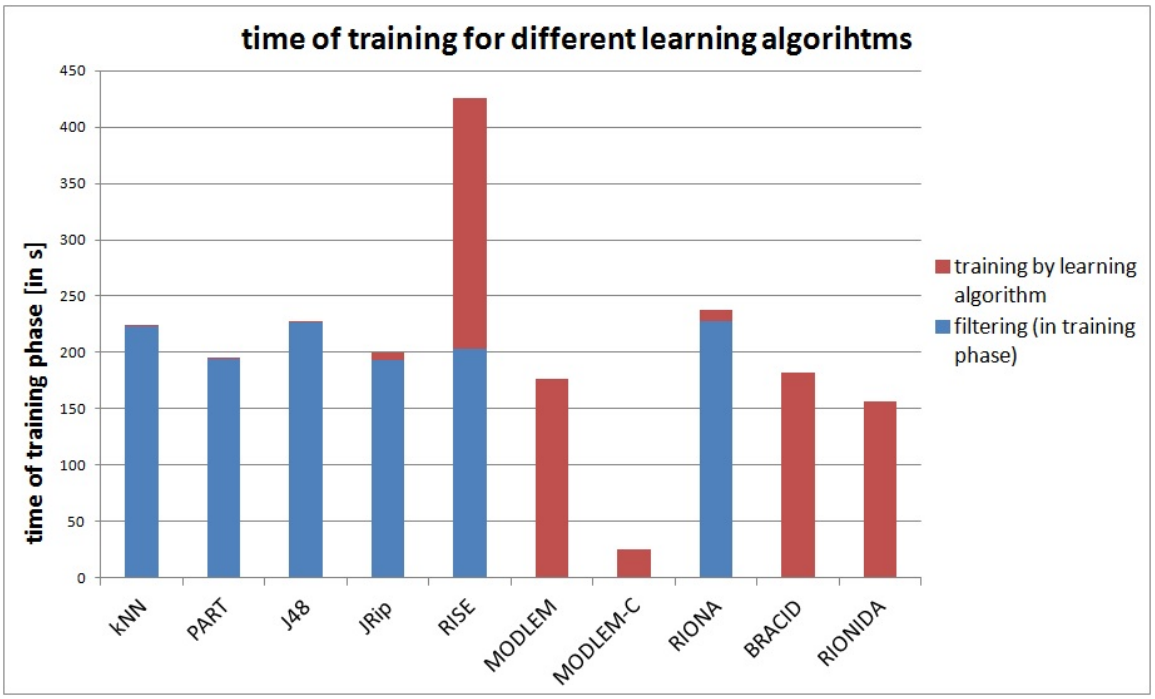


Figure 5.10: Summed (for all data sets used in experiments excluding *mammography*) average times of the training phase for single split in 10-fold stratified cross-validation (in seconds) for each learning algorithm used in experiments. In any split, the training set contains roughly 90% of the data set. Two times are distinguished: filtering (as the first part of the training phase), training by learning algorithm (as the second part of the training phase).

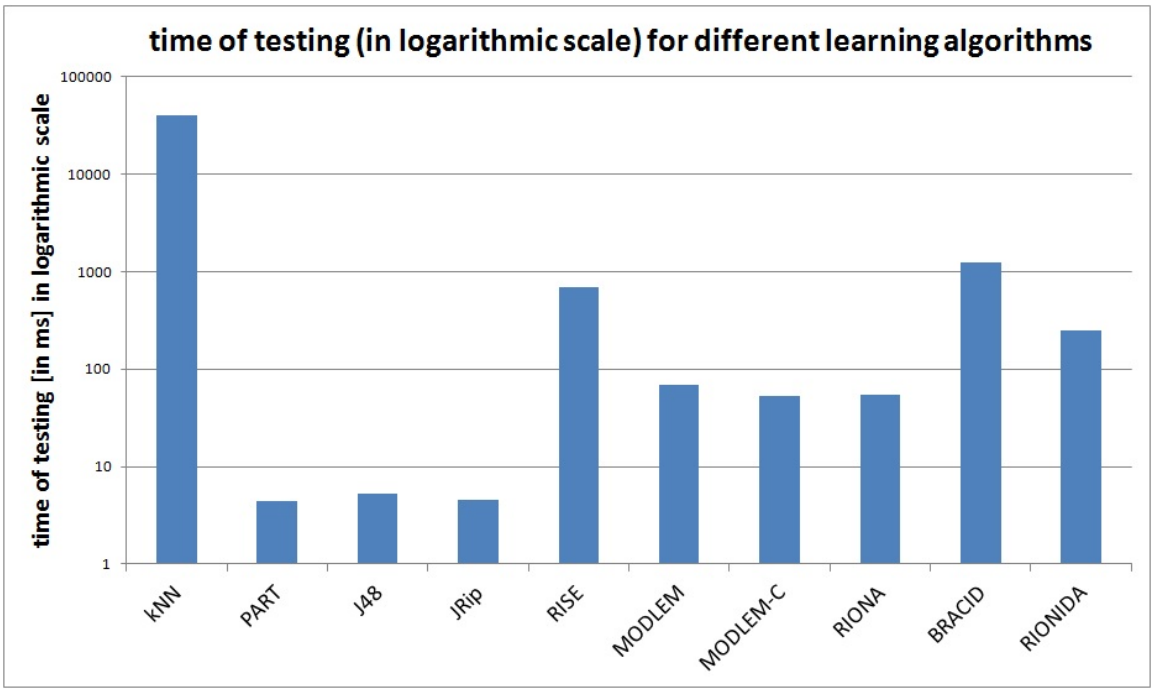


Figure 5.11: Summed (for all data sets used in experiments excluding *mammography*) average times of the testing phase for single split in 10-fold stratified cross-validation (in logarithmic scale, in milliseconds) for each learning algorithm used in experiments. In any split, the testing set contains roughly the one-tenth part of data set.

In Figure 5.11, we present time of testing phase for each learning algorithm (computed analogously as for training phase). RIONIDA is on the seventh place by means of testing time. It is around 50 times slower than PART, J48, and RIPPER algorithms; around 4 times slower than MODLEM, MODLEM-C, and RIONA. On the other hand, it is more than 150 times faster than kNN, and a few times faster than RISE and BRACID.

Taking into account that usually training phase is dominant, the RIONIDA algorithm on average has a comparable time of computations to the other AF-learners used in the experiments.

5.5 Additional experiments and their analysis

In this section, we present the results of some additional experiments. First, we analyse the use of RIONIDA, with filters dedicated to imbalanced data, in order to check whether RIONIDA realises the power of filters. Second, we present a more deep comparison of RIONIDA and RIONA (with different filters). Third, we present a more deep comparison of RIONIDA and BRACID. In the last three subsections, we present many additional experiments performed which possibly could lead to the improvement of RIONIDA quality. We used both the settings that are specific to RIONIDA (Subsection 5.5.4) and adopted from RIONA (Subsection 5.5.5). Moreover, we used (additionally implemented) modified versions of RIONIDA (Subsection 5.5.6).

In order to bound the length of the dissertation, we only present here the results in a compact way, without detailed analysis, e.g. statistical analysis. The aim is, in particular, to give the readers some intuition, whether some extensions or modifications in RIONIDA can improve its performance.

5.5.1 RIONIDA with filters

In the presented experiments, we applied learning algorithms dedicated to balanced data to the results of sampling methods (filters) dedicated to imbalanced data. We used two types of well-known filters. Let us recall that while designing comparative experiments, we assumed that such filters can not improve the quality of algorithms dedicated to imbalanced data (hence, such filters were not used in the mentioned context).

Here, we report the results of additional experiments checking whether this assumption actually holds in case of RIONIDA and BRACID. As an example, we present details of such a comparison for G-mean as the performance measure. We present detailed results of the combination of RIONIDA with all filters used in comparative experiments. Additionally to G-mean, we present Sensitivity and Specificity (i.e. the components of G-mean measure) obtained by RIONIDA without filters and with two filters used previously in the comparative experiments.

All these results are presented in Table 5.16. They show that for both filters, for all but one data set, the performance of RIONIDA worsens (corresponding to negative values in the table); in most cases by more than 1%. The worsening is mainly because of worsening of Sensitivity (in all cases for `SMOTE` and most cases for `SMOTE+ENN`). These results are understandable since by using filters, the algorithm is receiving as input the balanced training set, but the testing set is imbalanced.

The conclusion which follows from this experiment is that filters do not improve the quality of RIONIDA significantly. However, for example, such filters improved quality of BRACID for some data sets. Instead of presenting detailed experiments with resulting tables for BRACID, we present some exemplary results in this context. For `SMOTE+ENN` filter the obtained improvement (in G-mean) was greater than 1% for the following data sets: *abalone* (improvement of 3.71%), *cleveland* (5.42%), *ecoli* (1.38%), *glass* (26.35%), *yeast* (2.95%). For `SMOTE` filter the obtained improvement was greater than 1% for the following data sets: *abalone* (5.62%), *ecoli* (3.07%), *glass* (21.40%), *mammography* (4.55%), *yeast* (improvement of 1.07%). At the same time, as it was expected, for most data sets, the worsening was observed (both for `SMOTE+ENN` and `SMOTE`).

In the context of presented results, we can (roughly) conclude that RIONIDA realises the power of filters dedicated to imbalanced data and does much more for the relevant classification of imbalanced data.

5.5.2 Additional comparison of RIONIDA with RIONA

The analysis of comparative experiments in Section 5.3 showed that RIONIDA is significantly better than RIONA relative to G-mean as well as F-measure for all 3 considered strategies. Here, as an example, we present a more detailed comparison of

Table 5.16: The values of G-mean (in parenthesis Sensitivity, and Specificity) for RIONIDA$_G$ and the difference of these factors between RIONIDA$_G$ with different filters and RIONIDA$_G$ (without filter) for each data set used in experiments. These values are given in % and rounded to one decimal point. The changes above 1% are shown in bold. The changes below -1% are shown in bold and red. At the bottom, the averages of differences for all used data sets are also given.

	G-mean (Sensitivity, Specificity)		
	Values for	Differences between	
Data set	RIONIDA (with no filter)	RIONIDA with a filter and RIONIDA `SMOTE+ENN`	`SMOTE`
abalone	67.9 (67.8, 68.1)	**-8.0** (**-25.3**, **16.5**)	**-7.6** (**-23.7**, **14.3**)
balance-scale	77.0 (82.7, 71.8)	**-42.3** (**-68.4**, **12.9**)	**-64.7** (**-79.4**, **16.2**)
breast-cancer	65.0 (59.9, 70.5)	**-8.0** (**-9.5**, **-6.0**)	**-4.1** (**-8.6**, **1.8**)
breast-w	97.5 (98.6, 96.5)	0.2 (0.5, -0.1)	-0.5 (**-1.3**, 0.2)
car	96.7 (97.1, 96.4)	**-15.7** (**-28.7**, -0.2)	**-2.6** (**-4.8**, -0.4)
cleveland	76.4 (78.6, 74.3)	**-10.3** (**-23.7**, **5.6**)	**-19.3** (**-39.7**, **10.0**)
credit-g	69.9 (71.6, 68.3)	**-3.6** (**-1.7**, **-5.4**)	**-5.3** (**-15.5**, **6.2**)
ecoli	88.8 (89.7, 87.9)	**-2.2** (**-3.7**, -0.8)	**-7.1** (**-16.3**, **3.2**)
glass	69.3 (68.2, 70.5)	**-2.0** (**-15.3**, **15.2**)	**-3.2** (**-18.2**, **17.0**)
haberman	65.4 (68.9, 62.1)	**-5.3** (**1.6**, **-10.8**)	**-9.8** (**-19.6**, 0.7)
hepatitis	79.0 (78.1, 79.9)	**-5.0** (**-13.1**, **4.5**)	**-4.8** (**-13.8**, **5.8**)
ionosphere	90.9 (89.3, 92.5)	-0.5 (**-4.5**, **3.7**)	-0.4 (**-5.1**, **4.7**)
mammography	89.7 (85.1, 94.5)	**-14.8** (**-2.4**, **-26.8**)	-0.2 (**-2.2**, **2.0**)
new-thyroid	98.9 (99.1, 98.7)	0.0 (0.6, -0.6)	-0.5 (-0.9, -0.2)
nursery	99.9 (99.9, 99.9)	**-8.7** (**-15.9**, -0.9)	**-1.9** (**-3.0**, -0.8)
pima	72.9 (76.0, 69.9)	**-6.3** (**10.8**, **-18.9**)	**-5.7** (**-6.3**, **-5.3**)
postoperative	43.7 (39.2, 49.5)	**-10.0** (**-17.9**, **6.4**)	**-10.6** (**-20.0**, **8.6**)
transfusion	67.6 (66.1, 69.2)	**-4.4** (**6.6**, **-14.1**)	**-8.3** (**-11.2**, **-5.1**)
vehicle	95.1 (97.4, 92.9)	-0.6 (1.0, **-2.1**)	0.1 (**-2.1**, **2.2**)
yeast	85.0 (87.3, 82.7)	**-8.9** (**-25.9**, **11.5**)	**-7.0** (**-22.2**, **10.8**)
Averages of differences:		**-7.8** (**-11.8**, **-0.5**)	**-8.2** (**-15.7**, **4.6**)

these algorithms for G-mean performance measure. In this comparison, we present results of combining RIONA with all filters used in these experiments. Moreover, we use values of both Sensitivity and Specificity (i.e. the components of G-mean measure) obtained in comparative experiments by the considered algorithms. Certainly, the higher the values of Sensitivity and Specificity are, the better quality of the classifier is. In Table 5.17, all these results are presented.

Table 5.17: The values of G-mean (in parenthesis Sensitivity, and Specificity) for RIONIDA$_G$ and the difference of these factors between RIONA with different filters and RIONIDA$_G$ (for each data set used in experiments). These values are given in % and rounded to one decimal point. The changes above 1% are shown in bold. The changes below -1% are shown in bold and red. At the bottom, the averages of differences for all used data sets are also given.

	G-mean (Sensitivity, Specificity)			
	Values for	Differences between RIONA with a filter and RIONIDA (with no filter)		
Data set	**RIONIDA** (with **no filter**)	**RIONA** with **no filter**	**RIONA** with SMOTE+ENN	**RIONA** with SMOTE
abalone	67.9 (67.8, 68.1)	**-31.0** (**-54.1**, **31.4**)	**-8.0** (**-25.4**, **16.5**)	**-9.7** (**-27.7**, **16.3**)
balance-scale	77.0 (82.7, 71.8)	**-77.0** (**-82.7**, **28.2**)	**-43.7** (**-69.6**, **13.5**)	**-75.6** (**-82.4**, **17.9**)
breast-cancer	65.0 (59.9, 70.5)	**-11.6** (**-28.2**, **19.8**)	**-8.1** (**-8.9**, **-6.9**)	**-4.0** (**-6.7**, -0.5)
breast-w	97.5 (98.6, 96.4)	**-1.5** (**-3.7**, 0.7)	0.3 (0.4, 0.2)	-0.1 (-0.4, 0.2)
car	96.7 (97.1, 96.4)	**-11.4** (**-23.8**, **2.9**)	**-16.4** (**-29.9**, -0.2)	**-12.7** (**-24.6**, **1.2**)
cleveland	76.4 (78.6, 74.3)	**-76.4** (**-78.6**, **25.5**)	**-11.1** (**-25.4**, **6.1**)	**-20.6** (**-41.7**, **10.7**)
credit-g	69.9 (71.6, 68.3)	**-16.2** (**-40.5**, **24.7**)	**-3.5** (**-1.2**, **-5.7**)	**-5.4** (**-15.3**, **5.7**)
ecoli	88.8 (89.7, 87.9)	**-15.1** (**-33.7**, **9.3**)	**-2.1** (**-3.7**, -0.6)	**-7.5** (**-17.4**, **3.6**)
glass	69.3 (68.2, 70.5)	**-63.5** (**-66.5**, **28.2**)	**-2.5** (**-15.9**, **15.0**)	**-1.6** (**-15.9**, **17.3**)
haberman	65.4 (68.9, 62.1)	**-29.8** (**-55.1**, **29.7**)	**-5.6** (**4.1**, **-13.0**)	**-4.4** (**-3.5**, **-5.2**)
hepatitis	79.0 (78.1, 79.9)	**-17.6** (**-38.4**, **15.3**)	**-5.5** (**-14.4**, **4.8**)	**-6.9** (**-17.2**, **5.4**)
ionosphere	90.9 (89.3, 92.5)	**-3.6** (**-11.7**, **5.6**)	-0.5 (**-4.7**, **4.0**)	-0.3 (**-4.8**, **4.7**)
mammography	89.7 (85.1, 94.5)	**-17.6** (**-32.8**, **4.9**)	**-15.5** (**-4.3**, **-26.4**)	**-16.1** (**-5.5**, **-26.5**)
new-thyroid	98.9 (99.1, 98.7)	**-1.5** (**-3.1**, 0.2)	0.0 (0.6, -0.6)	-0.2 (-0.3, -0.2)
nursery	99.9 (99.9, 99.9)	-0.3 (-0.7, 0.0)	**-11.2** (**-20.5**, -0.7)	**-5.1** (**-9.5**, -0.4)
pima	72.9 (76.0, 69.9)	**-9.0** (**-31.2**, **21.1**)	**-6.3** (**10.8**, **-18.9**)	**-5.7** (**-6.3**, **-5.3**)
postoperative	43.7 (39.2, 49.5)	**-15.2** (**-27.9**, **34.1**)	**-9.6** (**-17.5**, **6.4**)	**-10.7** (**-20.4**, **8.9**)
transfusion	67.6 (66.1, 69.2)	**-18.2** (**-38.3**, **18.9**)	**-4.3** (**7.1**, **-14.4**)	**-9.1** (**1.5**, **-18.5**)
vehicle	95.1 (97.4, 92.9)	-0.8 (**-4.8**, **3.1**)	-0.6 (1.0, **-2.1**)	0.1 (**-2.1**, **2.2**)
yeast	85.0 (87.3, 82.7)	**-56.9** (**-79.2**, **17.1**)	**-9.0** (**-26.1**, **11.6**)	**-10.1** (**-28.2**, **12.3**)
Averages of differences:		**-23.7** (**-36.7**, **16.0**)	**-8.2** (**-12.2**, **-0.6**)	**-10.3** (**-16.4**, **2.5**)

From the presented results, some meaningful conclusions follow.

First, for RIONIDA, the factors Sensitivity and Specificity are quite close. This shows that RIONIDA is balancing well these two components of G-mean.

Second, for the three used filters (including the trivial one) G-mean for RIONA is worse than RIONIDA for all or almost all of the used data sets. This explains why RIONIDA outperforms RIONA in the above-mentioned main comparative experiment (even for the *maxG* strategy).

Third, for RIONA with no filter, Sensitivity is worse for all but one data set; however, Specificity is better for all data sets (for *nursery* it is, in fact, around 0.05, greater than 0%). Such a performance is typical for the algorithm dedicated to balanced data when they are used for imbalanced data, i.e. high Specificity with relatively low Sensitivity.

Fourth, the outperforming of RIONA by RIONIDA is not only related to better balancing between Sensitivity and Specificity by RIONIDA. For both filters `SMOTE` and `SMOTE+ENN`, for a few data sets, RIONA is worse than RIONIDA on Sensitivity as well as Specificity (this is discussed in more detail below). Also, for both filters `SMOTE` and `SMOTE+ENN`, for a few data sets, RIONA is worse than RIONIDA in terms of Sensitivity with Specificity almost unchanged. For RIONA with `SMOTE+ENN`, the averages of differences indicate that generally, Sensitivity is much better for RIONIDA than RIONA, but Specificity is only slightly better for RIONIDA than for RIONA. For RIONA with `SMOTE`, Specificity is on average better for RIONA than RIONIDA, but Sensitivity is on average much worse than for `SMOTE+ENN` filter.

Let us investigate more deeply the case of RIONA with `SMOTE+ENN` filter (the filter selected in the *optG* strategy for RIONA). In Figure 5.12, for all data sets used in experiments, the simultaneous difference (for these algorithms) of Sensitivity and Specificity is presented: ΔSens = RIONA Sensitivity − RIONIDA Sensitivity, ΔSpec = RIONA Specificity − RIONIDA Specificity. The negative (positive) value of x or y coordinate (for any data set represented by a point with these coordinates) denotes that RIONIDA achieved a higher (lower) value of Sensitivity or Specificity, respectively. From this figure (and Table 5.17) one can distinguish the following groups of cases taking into account whether RIONIDA outperforms (or not) RIONA (with `SMOTE+ENN` filter) by means of the considered factors:

A RIONIDA systematically outperforms RIONA: significantly or moderately on both Sensitivity and Specificity (for *mammography*, *breast-cancer*, *credit-g* data sets), significantly on Sensitivity and slightly on Specificity (for *car*, *nursery*, *ecoli*),

B RIONIDA significantly outperforms RIONA on Sensitivity with a significant loss on Specificity in such a way that the gain on Sensitivity is higher than the loss on Specificity, i.e. $|\Delta\text{Sens}| > |\Delta\text{Spec}|$ (for *balance-scale*, *cleveland*, *postoperative*, *yeast*, *abalone*, *hepatitis*),

C RIONIDA significantly outperforms RIONA on Specificity with a significant or moderate loss on Sensitivity in such a way that the gain on Specificity is higher than the loss on Sensitivity, i.e. $|\Delta\text{Spec}| > |\Delta\text{Sens}|$ (for *pima*, *haberman*, *transfusion*, *vehicle*),

D The values ΔSpec and $-\Delta$Sens are relatively close, i.e. RIONIDA outperforms RIONA on Sensitivity with the gain being similar to the loss on Specificity (for *glass*, *ionosphere*), or RIONIDA outperforms RIONA on Specificity with the gain being similar to the loss on Sensitivity (for *new-thyroid*),

A' RIONA slightly outperforms RIONIDA on both Sensitivity and Specificity (for *breast-w* data set),

where 'significant' means here that the absolute value is higher than 2%, 'moderate' – between 1%-2%, and 'slight' – less than 0.8%.

Group A is represented by points on the left of the y-axis and below the x-axis (the left bottom part of the figure). Group B is represented by points on the left of the y-axis and above the x-axis beyond points lying close to the line $y = -x$. Group C is represented by points on the right of the y-axis and below the x-axis beyond points lying close to the line $y = -x$. Group D is represented by points lying close to the line $y = -x$. Group A' is, in a sense, opposite to group A. It is represented by a single point lying close to the intersections of the x- and y-axes (slightly above and to the right).

For group A it is evident that RIONIDA outperforms RIONA on G-mean. For this group of data sets, RIONIDA outperforms RIONA relative to two criteria (i.e. simultaneously to Sensitivity and Specificity) of comparison. Also, for groups B and C it is relatively easy to explain (in terms of Sensitivity and Specificity) the fact that RIONIDA wins with RIONA. The informal explanation is that the gain on the one factor recovers the loss on the second factor. In the case of group D, the improvement for G-mean is obtained by acquiring to balance between Sensitivity and Specificity. Even for this group, RIONIDA outperforms RIONA on G-mean for two data sets (*glass* and *ionosphere*).

To sum up, for some data sets the improvement of RIONIDA relates to the improvement in both Sensitivity and Specificity. This especially shows the power of RIONIDA. For most of the analysed data sets, the improvement relates to proper balancing between Sensitivity and Specificity. In many cases worsening of one factor can lead to the much better improvement of the second factor.

All these observations confirm once again that it was far more successful to construct the RIONIDA algorithm than to use the RIONA algorithm with filters for imbalanced data.

The case of RIONA with different possible settings

In Section 3.6, we briefly described some extensions of the RIONA algorithm. One can ask whether for different than the default settings for RIONA the comparison of RIONIDA with RIONA can change. We checked the following settings (the default settings for RIONA used in the main comparative experiments are shown in bold):

1. voting method (***Equal***, *Inverse Square Distance*),
2. attribute weighting method (***None***, *Distanceased*, *Accuracy Based*),
3. distance measure (***City And Simplified Value Difference pseudoMetric***, *Interpolated Value Difference pseudoMetric*)

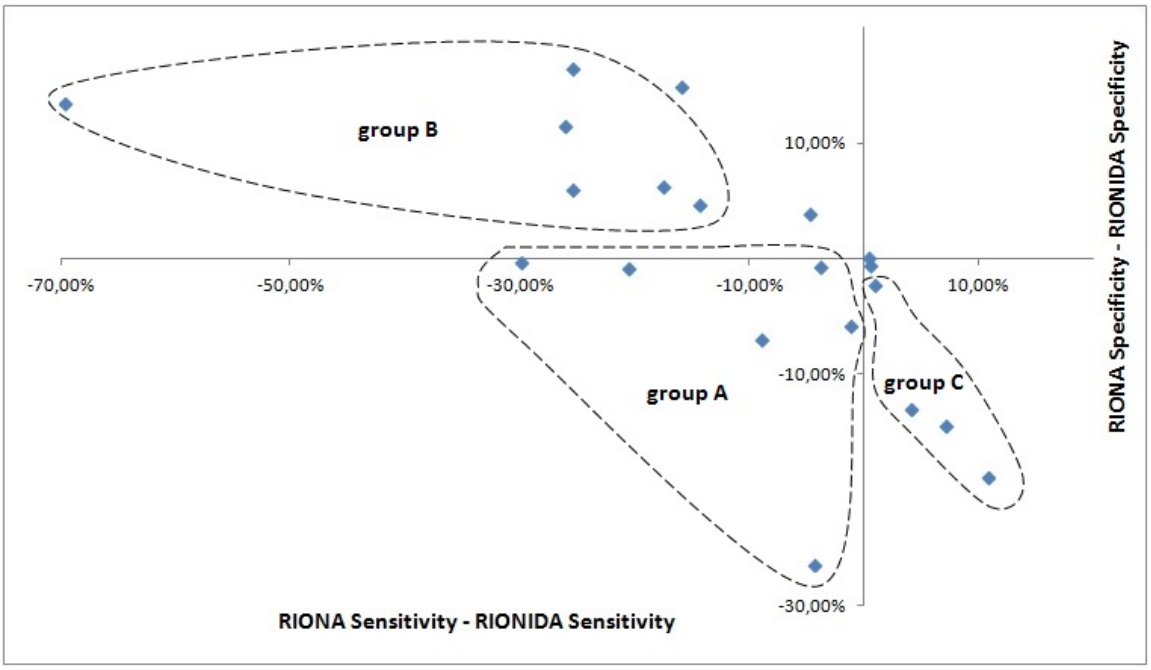


Figure 5.12: The difference in performance of RIONA (preceded with filter `SMOTE+ENN`) and RIONIDA for each data set used in experiments. Axes x and y represent the difference (for these algorithms) of Sensitivity and Specificity, respectively. Also, three of discussed (in this subsection) groups of cases are indicated (A, B, and C).

In Subsection 5.5.5, some comments on these settings are given with references where the more detailed description can be found.

We conducted preliminary experiments with all possible settings mentioned above, although not all data sets were used (*mammography* was excluded). Also, 3 possible filters were used for each combination. Thus, in total, it was $2{\cdot}3{\cdot}2{\cdot}3 = 36$ experiments. We do not present here the detailed results. However, the preliminary general conclusion was that none of these combinations leads to a significant improvement of the G-mean performance measure for RIONA. In particular, by taking maximal values of these 36 scores for each data set (as in the *max* strategy), only for 5 data sets (out of 19), these maximal values are higher for RIONA than for RIONIDA (for *breast-w*, *glass*, *new-thyroid*, *postoperative*, *vehicle*).

This shows that the fact of outperforming of RIONA by RIONIDA is not related to using not proper settings for RIONA. Thus, the conclusion given in the previous subsubsection can be strengthened that it was far more successful to construct the RIONIDA algorithm than to use the RIONA algorithm with specific settings and with filters for imbalanced data.

5.5.3 Additional comparison of RIONIDA with BRACID

In Section 5.3, it was shown that RIONIDA is significantly better than BRACID relative to G-mean as well as F-measure for all 3 considered strategies. We present a more detailed comparison of these algorithms for two performance measures: G-mean and F-measure.

The case of G-mean

Here, we present more detailed comparative results for RIONIDA$_G$ and BRACID. Moreover, we present an analysis of behaviour on the data sets of both Sensitivity and Specificity (i.e. the components of G-mean measure) obtained in comparative experiments by the considered algorithms. In Table 5.18, all these results are presented.

Table 5.18: The values of G-mean (in parenthesis Sensitivity, and Specificity) for RIONIDA$_G$ and the difference of these factors between BRACID and RIONIDA$_G$ for each data set used in experiments. These values are given in %. The changes above 1% are shown in bold. The changes below -1% are shown in bold and red. At the bottom, the averages of differences for all used data sets are also given.

	G-mean (Sensitivity, Specificity)	
	Values for	Differences between
Data set	**RIONIDA$_G$**	**BRACID** and RIONIDA$_G$
abalone	67.94 (67.82, 68.10)	**-2.14** (**-20.00**, **22.46**)
balance-scale	76.98 (82.65, 71.75)	**-18.30** (**-11.83**, **-22.88**)
breast-cancer	64.98 (59.88, 70.55)	**-6.81** (-0.59, **-13.38**)
breast-w	97.53 (98.63, 96.44)	-0.62 (0.41, **-1.62**)
car	96.74 (97.10, 96.39)	**-9.26** (**-19.42**, **2.17**)
cleveland	76.38 (78.57, 74.33)	**-13.49** (**-29.72**, **6.87**)
credit-g	69.90 (71.57, 68.27)	**-7.63** (**9.03**, **-20.14**)
ecoli	88.82 (89.71, 87.94)	**-4.40** (**-10.57**, **2.13**)
glass	69.26 (68.24, 70.51)	**-29.35** (**-51.18**, **24.88**)
haberman	65.40 (68.89, 62.13)	**-5.85** (**1.11**, **-11.33**)
hepatitis	79.00 (78.13, 79.92)	**-1.90** (**-4.06**, 0.41)
ionosphere	90.89 (89.29, 92.53)	0.52 (**7.70**, **-6.35**)
mammography	89.70 (85.12, 94.54)	**-4.29** (**-11.19**, **4.15**)
new-thyroid	98.93 (99.14, 98.72)	-0.24 (-0.86, 0.39)
nursery	99.90 (99.88, 99.92)	**-3.32** (**-6.52**, 0.00)
pima	72.87 (75.97, 69.92)	**-1.58** (**11.27**, **-11.66**)
postoperative	43.66 (39.17, 49.54)	**-1.17** (**14.17**, **-15.15**)
transfusion	67.64 (66.12, 69.21)	**-3.24** (**8.31**, **-13.49**)
vehicle	95.10 (97.39, 92.88)	**-1.28** (-0.96, **-1.59**)
yeast	84.95 (87.26, 82.74)	**-12.58** (**-32.75**, **13.43**)
Averages of differences:		**-6.35** (**-7.38**, **-2.04**)

For G-mean performance measure, RIONIDA$_G$ outperforms BRACID for all but one used data sets (for 2 data sets only slightly). The outperforming of BRACID by RIONIDA is not only related to better balancing between Sensitivity and Specificity by RIONIDA. For a few data sets, RIONIDA is better than BRACID in terms of Sensitivity as well as Specificity (this is discussed in more detail below). Also, the averages of differences indicate that generally, Sensitivity is much better for

RIONIDA than BRACID, but Specificity is only moderately better for RIONIDA than for BRACID.

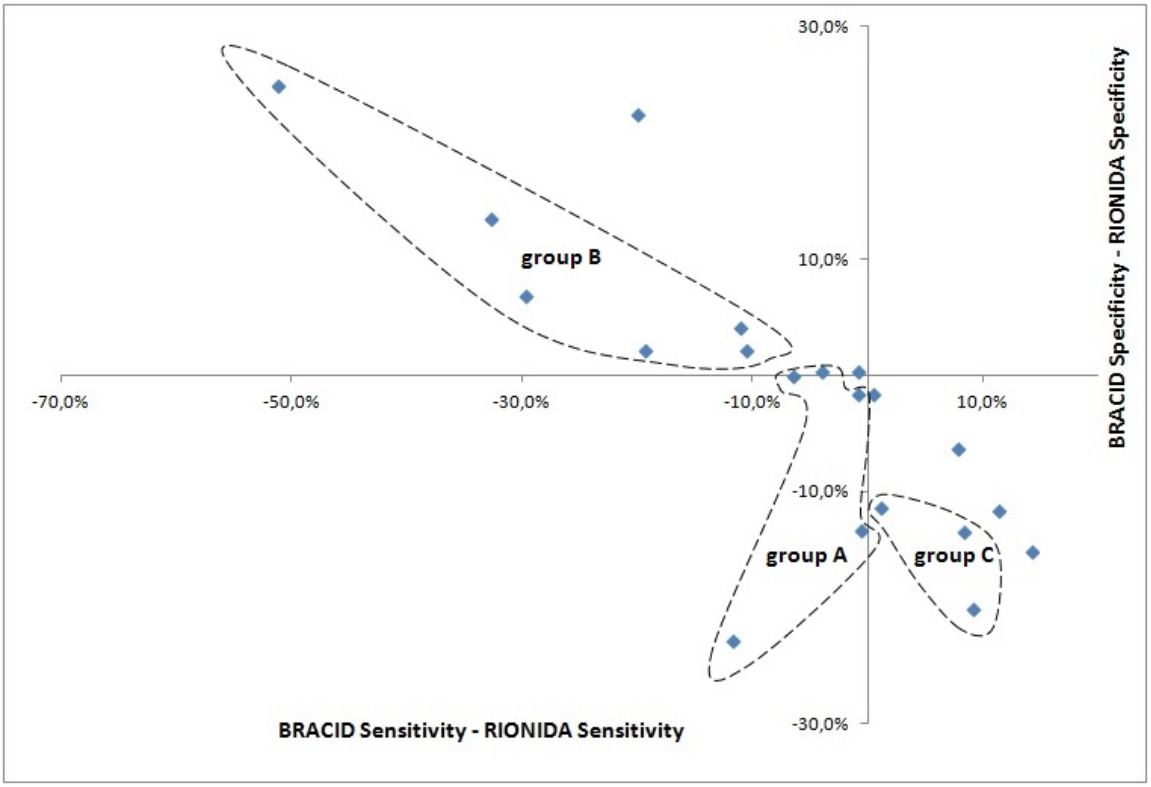


Figure 5.13: The difference in performance of BRACID and RIONIDA$_G$ for each data set used in experiments. Axes x and y represent the difference of Sensitivity and Specificity for these algorithms, respectively. Also, three of the discussed (in this subsection) groups of cases are distinguished (A, B, and C).

Let us discuss more deeply the differences between BRACID and RIONIDA analogously as it was done for RIONA and RIONIDA (in the previous subsection). In Figure 5.13, for each data set used in experiments, the simultaneous difference (for these algorithms) of Sensitivity and Specificity is presented: ΔSens = BRACID Sensitivity − RIONIDA Sensitivity; ΔSpec = BRACID Specificity − RIONIDA Specificity. The negative (positive) value of ΔSens denotes that RIONIDA outperforms BRACID (BRACID outperforms RIONIDA) on Sensitivity. The similar meaning of ΔSpec is for the relation between BRACID and RIONIDA on Specificity. From this figure (and Table 5.18) one can distinguish the following groups of cases (analogously as in the previous section) depending on whether RIONIDA outperforms (or not) BRACID by means of the considered factors:

A RIONIDA systematically outperforms BRACID: significantly on both Sensitivity and Specificity (for *balance-scale* data set), significantly on Specificity and slightly on Sensitivity (for *breast-cancer*), moderately on both Sensitivity and Specificity (for *vehicle*), significantly on Sensitivity with no difference on Specificity (for *nursery*), or significantly on Sensitivity with a slight loss on Specificity (for *hepatitis*),

B RIONIDA significantly outperforms BRACID on Sensitivity with a significant loss on Specificity in such a way that the gain on Sensitivity is higher than the loss on Specificity, i.e. $|\Delta\text{Sens}| > |\Delta\text{Spec}|$ (for *glass*, *cleveland*, *yeast*, *car*, *ecoli*, *mammography*),

C RIONIDA significantly outperforms BRACID on Specificity with a significant or moderate loss on Sensitivity in such a way that the gain on Specificity is higher than the loss on Sensitivity, i.e. $|\Delta\text{Spec}| > |\Delta\text{Sens}|$ (for *credit-g*, *haberman*, *transfusion*),

D The values ΔSpec and $-\Delta$Sens are relatively close, i.e. RIONIDA outperforms BRACID on Sensitivity with the gain being similar to the loss on Specificity (for *pima*, *postoperative*, *breast-w*, *ionosphere*), or RIONIDA outperforms BRACID on Specificity with the gain being similar to the loss on Sensitivity (for *abalone*, *new-thyroid*),

where 'significant' means here that the absolute value is higher than 2%, 'moderate' – between 0.9%-2%, 'slight' – less than 0.6%.

Groups A-D are analogous to the considered ones in the previous subsection with a small difference related to group A. Group A is represented by points on the left of the y-axis and below (or only slightly above) the x-axis (the left bottom part of the figure). This group represents data sets for which (except one – the case slightly above x-axis) RIONIDA outperforms BRACID in terms of both criteria (i.e. Sensitivity and Specificity) of comparison simultaneously. In the case of one data set from this group, such a relation is true only approximately.

Analogously as in the previous section, it is easy or relatively easy to explain in terms of Sensitivity and Specificity why RIONIDA outperforms BRACID on G-mean for data sets from groups A, B, and C.

However, even for 5 data sets from group D, RIONIDA outperforms BRACID on G-mean (moderately for 3 data sets, and slightly – for 2). This is most likely due to better balancing of Sensitivity versus Specificity by RIONIDA than BRACID.

To sum up, the sources for which RIONIDA wins with BRACID are as follows: for a group of data sets, RIONIDA outperforms BRACID on both components of G-mean (Sensitivity and Specificity); for another group, RIONIDA outperforms BRACID on one of these components gaining much more than losing on the second component; for other data sets, RIONIDA simply better balances between these components.

From all these observations, it follows that RIONIDA can reach relatively high values of primary components (Sensitivity, Specificity) for imbalanced data as well as maximises the given performance measure. It is worthy to note that RIONIDA and BRACID use the same CSVDM pseudometric, which previously proved to be successful for former algorithms, in particular for RIONA. Thus, the fact that RIONIDA outperforms BRACID is not related to the used pseudometric but has some deeper causes.

The case of F-measure

The BRACID algorithm is aiming to maximise F-measure (such an implementation is used in our experiments). Thus, exceptionally in this section, we present also details

related to comparative results for F-measure as the performance measure. Therefore, we present comparative results for RIONIDA$_F$ and BRACID. In Table 5.19, these results for F-measure are presented.

Table 5.19: The values of F-measure (in parenthesis Sensitivity, and Precision) for RIONIDA$_F$ and the difference of these factors between BRACID and RIONIDA$_F$ for each data set used in experiments. These values are given in %. The changes above 1% are shown in bold. The changes below -1% are shown in bold and red. At the bottom, the averages of differences for all used data sets are also given.

	F-measure (Sensitivity, Precision)	
	Values for	Differences between
Data set	**RIONIDA$_F$**	**BRACID** and RIONIDA$_F$
abalone	32.35 (38.09, 28.13)	**5.02** (**9.73**, **2.55**)
balance-scale	34.30 (63.47, 23.59)	**-15.96** (**7.35**, **-13.05**)
breast-cancer	52.17 (58.35, 47.21)	**-6.65** (0.94, **-10.22**)
breast-w	96.02 (98.59, 93.58)	**-1.18** (0.46, **-2.61**)
car	81.60 (82.61, 80.72)	**-8.41** (**-4.93**, **-11.46**)
cleveland	44.31 (70.29, 32.37)	**-10.95** (**-21.43**, **-7.03**)
credit-g	58.27 (72.34, 48.81)	**-4.83** (**8.27**, **-8.82**)
ecoli	68.36 (78.00, 60.94)	**-8.49** (**1.14**, **-12.77**)
glass	29.69 (41.18, 23.30)	**-9.97** (**-24.12**, 0.51)
haberman	49.70 (69.63, 38.67)	**-4.09** (0.37, **-4.82**)
hepatitis	60.31 (70.00, 53.07)	-0.93 (**4.06**, **-3.44**)
ionosphere	87.38 (86.03, 88.81)	0.14 (**10.96**, **-9.04**)
mammography	67.33 (63.81, 71.28)	**-2.76** (**10.11**, **-13.95**)
new-thyroid	96.40 (99.14, 93.84)	0.51 (-0.86, **1.80**)
nursery	98.92 (99.30, 98.56)	**-3.82** (**-5.95**, **-1.62**)
pima	66.04 (80.22, 56.15)	-0.23 (**7.01**, **-3.31**)
postoperative	33.61 (61.67, 23.16)	**-1.68** (**-8.33**, -0.32)
transfusion	50.02 (58.37, 43.85)	**-2.93** (**16.07**, **-9.41**)
vehicle	89.79 (92.56, 87.19)	**-3.98** (**3.87**, **-9.89**)
yeast	41.24 (40.59, 42.10)	0.38 (**13.92**, **-8.39**)
Averages of differences:	**-4.04** (**1.43**, **-6.27**)	

In Figure 5.14, for all data sets used in experiments, simultaneously the difference of values of Sensitivity[13] and Precision (i.e. the components of F-measure) for these algorithms are presented: ΔSens = BRACID Sensitivity − RIONIDA Sensitivity; ΔPrec = BRACID Precision − RIONIDA Precision. From this figure (and Table 5.19) one can observe that for all but three data sets, RIONIDA has much better Precision than BRACID. On the other hand, for most data sets, BRACID has better Sensitivity. From this figure (and Table 5.19) one can distinguish the following groups of cases (analogously to the previous section) taking into account whether RIONIDA outperforms (or not) BRACID by means of the considered factors:

[13] In such context Sensitivity is usually called Recall (see Subsection 2.6.1).

A RIONIDA systematically outperforms BRACID: significantly or moderately on both Sensitivity and Precision (for *cleveland*, *car*, *nursery*), significantly on Precision and slightly on Sensitivity (for *postoperative*), significantly on Sensitivity with a slight loss on Precision (for *glass*), or significantly on Precision with a slight loss on Sensitivity (for *haberman*, *breast-w*),

C RIONIDA significantly outperforms BRACID on Precision with a significant or moderate loss on Sensitivity in such a way that the gain on Precision is higher than the loss on Sensitivity, i.e. $|\Delta\text{Prec}| > |\Delta\text{Sens}|$ (for *balance-scale*, *ecoli*, *breast-cancer*, *vehicle*, *mammography*),

D The values ΔPrec and $-\Delta$Sens are relatively close, i.e. RIONIDA outperforms BRACID on Sensitivity with the gain being similar to the loss on Precision (for *new-thyroid*), or RIONIDA outperforms BRACID on Precision with the gain being similar to the loss on Sensitivity (for *credit-g*, *hepatitis*),

A' BRACID significantly outperforms RIONIDA on both Sensitivity and Precision (for *abalone*),

C' BRACID significantly outperforms RIONIDA on Sensitivity with a significant loss on Precision in such a way that the gain on Sensitivity is higher than the loss on Precision, i.e. $|\Delta\text{Sens}| > |\Delta\text{Prec}|$ (for *transfusion*, *pima*, *ionosphere*, *yeast*),

where 'significant' means here that the absolute value is higher than 2%, 'moderate' – between 0.9%-2%, and 'slight' – less than 0.6%.

Again, group A represents data sets for which RIONIDA wins (for 3 data sets roughly wins) with BRACID using simultaneously two criteria of comparison on Sensitivity and Precision. Analogously as in the previous sections, it is easy or relatively easy to explain in terms of Sensitivity and Precision why RIONIDA outperforms BRACID on F-measure for data sets from groups A, and C.

Groups A' and C' are, in a sense, opposite to groups A and C, respectively. The one data set (*abalone*) in group A' is the only one data set for which RIONIDA performs worse than BRACID significantly. Probably the fact of weak results of RIONIDA on *abalone* data set relates to different borderline regions with different distributions of the minority and majority classes. We have an idea of how to improve the performance of RIONIDA for such cases (see Section 6.2).

If we consider group C', which consists of 4 data sets, BRACID: slightly wins with (RIONIDA) on 2 of these data sets (*yeast*, *ionosphere*), slightly loses on 1 data set (*pima*), and significantly loses on 1 data set (*transfusion*). It seems surprising why RIONIDA wins with BRACID in case of *transfusion* data set (with such differences of Sensitivity and Precision for BRACID and RIONIDA). This occurs probably because in this case BRACID does not balance well Sensitivity and Precision (74.44% and 34.44%), while RIONIDA balances it better (58.37% and 43.85%). It is an example showing that RIONIDA can adapt quite well to the chosen performance measure.

To sum up, the sources for which RIONIDA wins with BRACID for F-measure are similar to those mentioned in the previous subsubsection for G-mean. Additionally, it is worth underlining that even for the group of data sets being natural candidates

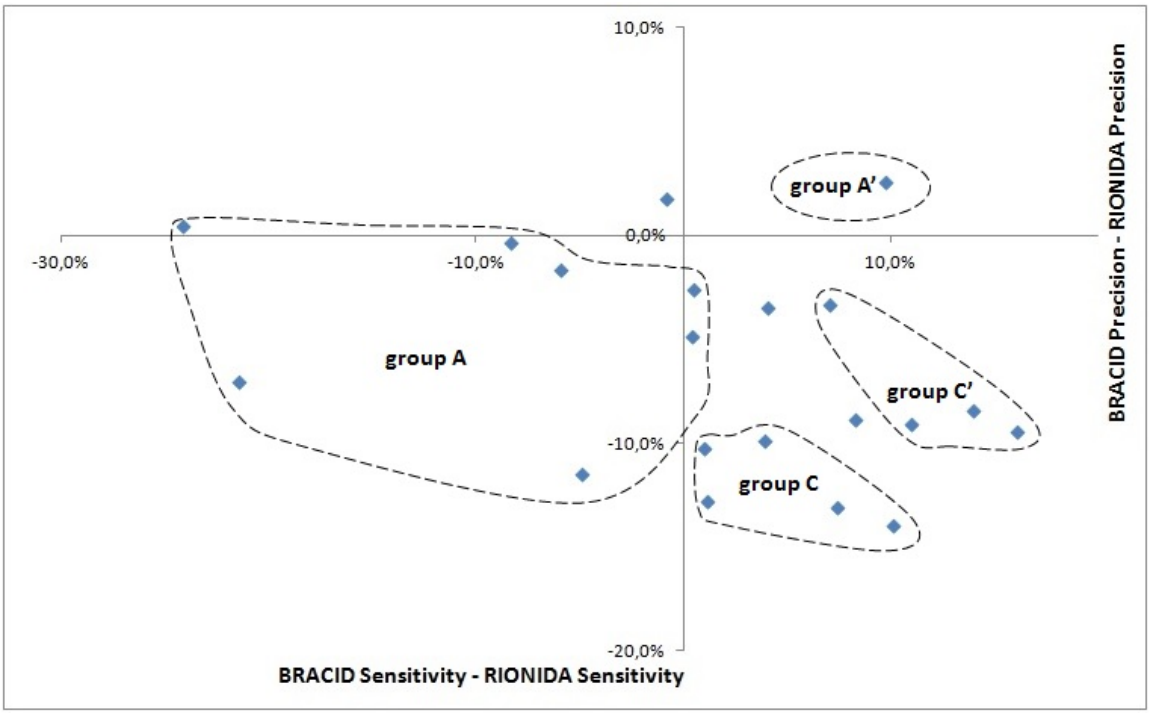


Figure 5.14: The difference in performance of BRACID and RIONIDA$_{\mathrm{F}}$ for each data set used in experiments. Axes x and y represent the difference (for these algorithms) of Sensitivity and Precision, respectively. Also, four of the discussed (in this subsection) groups of cases are distinguished (A, C, A', and C').

for RIONIDA to lose (group C'), still it achieves relatively good results. Again this shows that RIONIDA balances components of the considered performance measure adequately.

Thus, the conclusion given in the previous subsubsection can be strengthened that RIONIDA can reach relatively high values of the primary components for imbalanced data (Sensitivity, Specificity, Precision) as well as maximise the given performance measure. All this makes RIONIDA competitive with algorithms dedicated to imbalanced data (such as BRACID).

5.5.4 The RIONIDA quality analysis for different settings specific to RIONIDA

In this and the subsequent section, we analyse (informally) whether the RIONIDA quality can be improved by altering the RIONIDA settings. In this section, we present the results of changing some of the settings of parameters which are specifically associated with the design of RIONIDA. In the next section, we deal with parameters of RIONIDA adopted to RIONIDA from RIONA.

As it was mentioned in Subsection 4.4.2, we use the following default settings for RIONIDA:

- $K = K_{def}$, where $K_{def} = \{1, 2, \ldots, 100\}$ (which means that $k_{max} = |K| = 100$);
- $P = P_{def}$, where $P_{def} = \{0.00, 0.01, 0.02, \ldots 0.5\}$;

- $S = S_{def}$, where $S_{def} = \{-0.1, 0.0, 0.1, \ldots, 1.0\}$.

Three groups of additional comparative experiments were performed to check whether the changes in these default settings of RIONIDA can improve its performance.

We performed these comparative experiments only for the performance measure G-mean. Thus, we used $\text{RIONIDA}_{\text{G}}$; and therefore, in the current analysis, RIONIDA will refer to this setting.

To save space, we use one table for presenting these experiments, i.e. Table 5.20. We show in this table: scores of RIONIDA (presented earlier) for its default setting; and for each of the considered settings, the difference between scores for such setting and the default setting. Thus, positive values denote improvement of RIONIDA performance (G-mean), and negative values denote its worsening.

Table 5.20: The values of G-mean (in %) for RIONIDA$_{\mathrm{G}}$ (with default settings) and the change of scores (in %) for different modifications of default settings of RIONIDA$_{\mathrm{G}}$ (for different data sets). The following parameters are examined with non-standard settings: k_{max} (= $|K|$, the size of the neighbourhood to be used during searching for the optimal neighbourhood size), P (the set of admissible values of parameter p), S (the set of admissible values of parameter s). The values of the parameters used in the table can be found in Subsection 5.5.4. The changes above 1% are shown in bold. The changes below -1% are shown in bold and red. At the bottom, the averages of differences for all used data sets are also given.

	Score for	Differences between scores for different RIONIDA settings and the default setting for							
Data set	**default**	$k_{max} = 200$	$P = P_1$	$S = S_1$	$S = S_2$	$S = S_3$	$S = S_4$	$S = S_5$	$S = S_6$
abalone	67.94	0.61	0.14	-0.04	-0.03	-0.03	-0.05	-0.05	-0.05
balance-scale	76.98	0.00	0.03	**-35.41**	**-3.65**	**-3.65**	**-3.65**	**-3.65**	**-3.65**
breast-cancer	64.98	0.01	0.14	**-1.56**	**1.15**	0.13	-0.61	0.61	-0.78
breast-w	97.53	-0.07	0.01	0.02	0.03	0.00	-0.01	0.00	0.00
car	96.74	0.02	-0.08	**-1.68**	-0.33	0.20	-0.43	-0.90	-0.36
cleveland	76.38	0.24	-0.28	0.04	-0.08	-0.08	0.00	0.00	0.00
credit-g	69.90	0.31	0.08	0.05	0.06	0.06	-0.01	-0.01	-0.01
ecoli	88.82	-0.23	0.07	0.18	0.27	0.27	0.17	0.17	0.17
glass	69.26	0.00	0.00	0.59	0.13	-0.44	0.00	0.53	-0.03
haberman	65.40	0.51	-0.05	-0.38	**-4.01**	**-3.98**	**-2.17**	**-1.49**	**-2.33**
hepatitis	79.00	0.00	-0.15	**-1.99**	0.00	0.00	0.00	0.00	0.00
ionosphere	90.89	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
mammography	89.70	-0.10	0.16	-0.29	-0.27	-0.27	-0.42	-0.42	-0.42
new-thyroid	98.93	0.00	-0.17	-0.31	0.00	0.00	0.00	0.00	0.00
nursery	99.90	0.03	0.00	-0.02	-0.18	-0.16	0.00	-0.01	0.00
pima	72.87	-0.47	-0.17	-0.12	-0.06	-0.06	-0.16	-0.16	-0.16
postoperative	43.66	0.00	-0.08	**3.26**	0.84	-0.16	**-1.02**	**1.97**	-0.63
transfusion	67.64	-0.14	0.02	**-4.18**	**-3.39**	0.00	0.00	**-3.52**	0.00
vehicle	95.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
yeast	84.95	-0.49	-0.14	-0.08	-0.71	-0.73	-0.08	-0.08	-0.08
Averages of differences:		**0.01**	**-0.02**	**-2.10**	**-0.51**	**-0.45**	**-0.42**	**-0.35**	**-0.42**

Different maximal k value

The RIONIDA algorithm uses $k_{max} = |K| = 100$ as default. We tried to examine whether this value is not too small. We experimented with $k_{max} = |K| = 200$.

Results in Table 5.20 (column for $k_{max} = 200$) show that differences for all data sets are below 1% (even below 0.5% for all but 2 data sets) with very slight general improvement (average of differences) in favour of the setting $k_{max} = 200$. We can conclude that setting $k_{max} = |K| = 200$ did not significantly improve the performance of the RIONIDA and the default setting for k_{max} is satisfactory.

Let us recall that it was experimentally checked for the RIONA algorithm that there is no need to use the whole training set in the process of classification and classifier learning. Moreover, the bound of the neighbourhood size can even improve the classification performance or at least not reduce it significantly. The development of RIONIDA was based on the assumption that this (experimentally checked) hypothesis can be extended for the RIONIDA with different performance measures than Accuracy (see Subsection 4.4.2). The experiment, presented here, can be treated as an argument for such a hypothesis. However, more experiments should be performed as in the case of the RIONA algorithm to check (experimentally) this pre-assumed hypothesis more thoroughly (see Section 6.2).

Different sets of admissible values for parameter p

The RIONIDA algorithm uses by default $P = P_{def}$. We tried to check experimentally whether a more detailed scale can improve the performance of RIONIDA.

We experimented with $P = P_1$, where $P_1 = \{0.0, 0.001, 0.002, ..., 0.499, 0.500\}$.

Results in Table 5.20 (column for $P = P_1$) show that differences for all data sets are below 1% with very slight general worsening (negative average of differences) in comparison with the default setting. We can conclude that there is no evidence that setting $P = P_1$ can improve the performance of RIONIDA in comparison with the default setting.

Different settings of parameter s

We tried to examine whether using set S (of admissible values for the parameter s) with different sets than the default one (S_{def}) can improve the performance of RIONIDA. In each experiment, we used one of the following settings: $S = S_1$, $S = S_2$, $S = S_3$, $S = S_4$, $S = S_5$, or $S = S_6$, where:

- $S_1 = \{1.0\}$,
- $S_2 = \{0.0\}$,
- $S_3 = \{-0.1\}$,
- $S_4 = \{-0.1, 1.0\}$,
- $S_5 = \{0.0, 1.0\}$,
- $S_6 = \{-0.1, 0.0, 1.0\}$.

Let us recall (see Subsection 4.3.5) that the case with $S = S_3 = \{-0.1\}$ corresponds to the situation when we do not check the consistency of examples in RIONIDA (it corresponds to the kNN method with the optimal neighbourhood and the optimal value of the parameter p). The case with $S_1 = \{1.0\}$ represents the RIONIDA algorithm with pure rules. For data sets with no inconsistencies, the case with $S_2 = \{0.0\}$ is equivalent to the case with $S_3 = \{-0.1\}$.

Results in Table 5.20 (columns for $S = S_1, \ldots, S = S_6$) show that reducing the default set to one of the used sets generally worsens the performance of RIONIDA. Only for three of these settings, one can observe improvement of more than 1% for one data set in each case. However, in these three cases, the performance worsens by more than 1% for (at least) three other data sets. Also, the averages of differences indicate the negative outcome for these different settings.

Moreover, this experiment shows us a few other noteworthy things. First, when pure rules are used (case $S = S_1$), the performance for the (mentioned previously) *balance-scale* data set worsens very significantly (and also worsens for a few other data sets). Second, using RIONIDA with no consistency checking (case $S = S_3$, which corresponds to the instance-based method like kNN) generally worsens the performance of RIONIDA. Third, cases for $S = S_2$ and $S = S_3$ do not give the same results. This is due to the existing inconsistencies in data sets. The differences in performance for these two settings show that using such separate settings (e.g. in the process of learning of the optimal values) can be profitable. Fourth, only learning whether to use either pure rules or instance-based (kNN like) classifier (cases $S = S_4$, $S = S_5$, $S = S_6$, which represent small alterations of the required type of optimisation) can be not sufficient to find the classifier with high performance.

Thus, the performed experiments can be treated as another argument beyond those already given in Subsection 4.3.5 that it is reasonable to introduce the additional parameter s and to search the space of its admissible values at the learning step (aiming at tuning levels of rules inconsistency) in RIONIDA. The evident example for this is the previously discussed *balance-scale* data set.

5.5.5 The RIONIDA quality analysis for different RIONIDA settings adopted from RIONA

In Section 3.6, we briefly described some extensions of the RIONA algorithm. Since RIONIDA was implemented based on RIONA, all the parameters of RIONA were adopted into RIONIDA. For a few of these parameters, it makes sense to use them also in RIONIDA together with an attempt to find different settings than the default. These are the following (the default settings for RIONIDA are shown in bold):

1. voting method (***Equal***, *Inverse Distance*, *Inverse Square Distance*),
2. attribute weighting method (***None***, *Perceptron*, *Distance Based*, *Accuracy Based*),
3. distance measure (*City And Hamming Metric*, ***City And Simplified Value Difference pseudoMetric***, *Density Based Value Difference Metric*, *Interpolated Value Difference pseudoMetric*)

In Subsection 3.6.2, one can find how *Equal* (generally used in the thesis), *Inverse Distance* and *Inverse Square Distance* voting types are used.

In Subsection 3.6.3, one can find the general idea of using different weights for attributes. Generally, in the thesis, equal weights are used (*None* setting). How to count weights related to settings *Perceptron*, *Distance Based*, *Accuracy Based* is not described in the thesis, but it can be found in [219].

Let us recall here that *City And Simplified Value Difference pseudoMetric* is the metric[14] generally used in the thesis (see Subsection 2.2.3). The *City And Hamming Metric* is a quite basic metric also described in Subsection 2.2.3. The ideas of *Density Based Value Difference Metric* and *Interpolated Value Difference Metric* are briefly described in Subsection 3.6.4 (for details see [219]).

We conducted preliminary experiments with all possible settings mentioned above, although not all data sets were used (*mammography*, *glass*, and *new-thyroid* data sets were excluded). Thus, in total, it was $3 \cdot 4 \cdot 4 = 48$ experiments. The preliminary general conclusion was that none of these combinations leads to the improvement of the G-mean performance measure.

We repeated these experiments for all data sets used in previous comparative experiments, although with the selected, limited number of settings. We present in Table 5.21 results of experiments obtained by changing one of these settings and other setting being fixed with their default values of RIONIDA (in addition to the default settings, three groups consisting of 2, 3, and 3 experiments).

[14]For simplicity we do not distinguish between metrics and pseudometrics in this subsubsection (unless indicated in the name).

Table 5.21: The values of G-mean (in %) for different data sets for RIONIDA$_G$ and the change of scores (in %) for different modifications of default settings of RIONIDA$_G$, settings adopted from RIONA. We use a single letter for denoting a parameter: V (voting method), W (attribute weighting method), or M (distance measure). The values of the parameters are listed at the beginning of Subsection 5.5.5. As the abbreviations of these values, the capital letters from the name are used. For example, V=*ISD* denotes the following setting: voting method set to *Inverse Square Distance*, and other default settings of RIONIDA (presented at the beginning of Subsection 5.5.5). The changes above 1% are shown in bold. The changes below -1% are shown in bold and red. At the bottom, the averages of differences for all used data sets are also given.

	Score for	Differences between scores for different settings and the default setting for							
Data set	**default**	**V=*ID***	**V=*ISD***	**W=*P***	**W=*DB***	**W=*AB***	**M=*CAH***	**M=*DBVD***	**M=*IVD***
abalone	67.94	0.05	-0.83	0.18	0.36	0.29	0.31	**6.14**	-0.13
balance-scale	76.98	**-6.65**	**-6.72**	**-4.32**	**-5.91**	0.00	0.00	**-22.03**	**-3.83**
breast-cancer	64.98	**-1.38**	**-2.93**	-0.36	**-2.47**	0.11	-0.79	0.00	0.00
breast-w	97.53	0.08	-0.03	-0.17	-0.53	0.02	-0.16	-0.22	-0.38
car	96.74	-0.91	**-2.52**	0.16	0.48	-0.56	-0.82	0.00	0.00
cleveland	76.38	**1.81**	**1.18**	**1.23**	-0.42	0.76	**1.17**	-0.04	-0.53
credit-g	69.90	0.41	0.06	-0.02	0.16	**-1.65**	**-1.59**	0.44	0.56
ecoli	88.82	-0.57	-0.79	-0.18	-0.68	0.07	-0.39	**-1.30**	**-2.13**
glass	69.26	**2.62**	**4.00**	**1.04**	**2.83**	**-7.96**	0.00	**-69.26**	**-7.14**
haberman	65.40	**-2.94**	**-3.70**	-0.29	-0.39	-0.21	0.24	**-3.29**	**-8.39**
hepatitis	79.00	0.27	-0.63	-0.44	**-2.51**	-0.69	0.19	**-2.41**	**1.56**
ionosphere	90.89	-0.21	-0.03	-0.69	0.00	0.06	0.00	**-1.69**	**-2.78**
mammography	89.70	0.17	0.24	0.31	-0.36	0.00	0.00	**-2.23**	**-1.22**
new-thyroid	98.93	0.09	-0.29	-0.17	**-1.38**	0.00	0.00	**-98.93**	**-2.91**
nursery	99.90	-0.13	-0.10	0.00	0.01	-0.12	-0.04	0.00	0.00
pima	72.87	-0.02	0.24	0.03	**1.52**	**-3.86**	-0.04	**2.03**	0.82
postoperative	43.66	**1.80**	**3.53**	**1.19**	**4.00**	0.06	**-4.40**	0.00	0.00
transfusion	67.64	**-4.54**	**-7.14**	-0.19	**-4.19**	**-1.23**	0.19	**-1.13**	**-6.70**
vehicle	95.10	0.38	0.77	0.05	**1.59**	0.10	0.00	**-1.09**	**-8.19**
yeast	84.95	-0.10	-0.74	0.22	**-2.19**	0.00	0.00	-0.85	**-4.52**
Averages of differences:		**-0.49**	**-0.82**	**-0.12**	**-0.50**	**-0.74**	**-0.31**	**-9.79**	**-2.29**

Different voting methods

The RIONIDA algorithm uses by default equal voting irrespective of the distance of the training object from the considered test object.

We also experimented with two other voting methods depending on the distance of the training object to be used for voting from the considered test object (*Inverse Distance* or *Inverse Square Distance*).

Results in Table 5.21 (column with the prefix 'V=') show that for both of these methods, for three data sets one can observe improvement by more than 1%; and for 4 or 5 data sets one can observe worsening by more than 1%. Also, the negative averages of differences indicate the negative outcome for these settings. One can (roughly) conclude that for both additional methods of voting the performance of RIONIDA worsens.

However, it seems incomprehensible that the voting method taking into account the distance of the training object from the test one does not lead to significant improvement here. For the RIONA algorithm, it showed an improvement for many data sets (see [219]). Probably for using these voting methods, one should choose the scale for the parameter p in a different way (different setting of P). This seems worthy to check in future experiments (see Section 6.2).

Different attribute weighting methods

The RIONIDA algorithm uses by default equal weights for all attributes (all attributes are treated as equally important).

We also experimented with three other methods assigning relevant weights for particular attributes (attribute weighting method as *Perceptron*, *Distance Based*, and *Accuracy Based*).

Results in Table 5.21 (column with the prefix 'W=') show that for the *Perceptron* method, for 3 data sets, one can observe improvement by slightly more than 1%; and for 1 data set, one can observe worsening by more than 4%. The average of differences is equal to -0.12%. Hence, we can (roughly) conclude that this method does not bring significant improvement. However, it shows signs that it could be beneficial to investigate this method in future research. Probably one can make this method more competitive by simultaneously choosing the appropriate scale for the parameter p (analogously as mentioned in the previous subsubsection). This seems worthy to check in future experiments (see Section 6.2).

For the *Distance Based* method, for 4 data sets, one can observe improvement by more than 1%; and for 6 data sets worsening by more than 1%. The average of differences is equal to -0.5%. Hence, one can (roughly) conclude that this method used in RIONIDA worsens its performance.

For the *Accuracy Based* method, one can observe significant worsening of the performance of RIONIDA: for 4 data sets the worsening by more than 1%, and the average of differences is around -0.7%. However, these results seem justifiable since this method directly tries to maximise Accuracy performance measure, generally not satisfactory for the imbalanced learning problem.

Different distance measure

The RIONIDA algorithm uses by default *City And Simplified Value Difference pseudoMetric*.

We also experimented with three other methods for calculating distance measure (*City And Hamming Metric*, *Density Based Value Difference Metric*, and *Interpolated Value Difference Metric*).

Results in Table 5.21 (column with the prefix 'M=') show that for the two last of these methods the general performance of RIONIDA significantly worsens: for both methods for 10 data sets the quality worsens by more than 1% (a few times even far more), improves by more than 1% only for 1 or 2 data sets, and the average differences are considerably negative. Such an outcome can seem surprising since these distance measures are more compound than the default one. However, these results coincide with the results presented in [219]. It was shown there that these measures lead to the general worsening of the performance (of Accuracy) for balanced data sets. One of the reasons for that can be the fact mentioned in Subsection 3.6.4 that by using these metrics inconsistent local rules can be recognised as consistent in both RIONA and RIONIDA.

For the *City And Hamming Metric* method, one can observe improvement by more than 1% in case of one data set; and worsening by more than 1% in case of two data sets (also worsening by about 1% in case of two other data sets). The average of differences is equal to around -0.3%. It is clear that for data sets with no nominal attributes this method gives similar or identical results to the default one (we have 11 such data sets, i.e. *breast-w*, *ecoli*, *glass*, *haberman*, *ionosphere*, *mammography*, *new-thyroid*, *pima*, *transfusion*, *vehicle*, *yeast*). It is understandable that for the remaining data sets the general improvement is observed since the default method makes more informative use of relations between values of symbolic attributes than the considered one here.

5.5.6 The RIONIDA quality analysis for different extended versions of RIONIDA

Based on RIONIDA (presented in Chapter 4), some extended ideas of this algorithm were developed. We temporarily implemented these extended versions of RIONIDA to check whether it can be beneficial to invest these directions in future. We also implemented a version of RIONIDA to check whether a greater portion of data can lead to finding significantly better optimal values of internal parameters.

In this section, results of three groups of experiments related to three implemented extensions of RIONIDA are presented.

In Table 5.22, the summary of experiments for these three groups of experiments is presented. We show in this table: scores of RIONIDA for its standard implementation with default settings (presented earlier), and the difference between scores of the new implementation with considered settings and the standard implementation with default settings. Thus, positive values denote improvement of RIONIDA performance (G-mean), and negative values denote its worsening.

Table 5.22: The values of G-mean (in %) for different data sets for the standard implementation with default settings of RIONIDA with the optimisation measure set to G-mean (hence, $\text{RIONIDA}_{\text{G}}$ is used) and the change of scores (in %) for different extensions of RIONIDA. The 3 groups of scores for 3 extended implementations of RIONIDA are presented: optimisation by the stratified cross-validation (CV opt.), optimisation in 4-dimensional parameter space (4D), and optimisation of parameters with larger training data sets (Optimal). In the case of 4D implementation, 5 different settings were checked in experiments. In the case of Optimal implementation, this is, in fact, a specific usage of RIONIDA (not typical for learning algorithm). The changes above 1% are shown in bold. The changes below -1% are shown in bold and red. At the bottom, the averages of differences for all used data sets are also given.

Implementation: / Setting: / Data set	Score for **standard default**	Differences between scores for different implementations and the standard implementation for **CV opt. default**	**4D** $S_{min} = S_{def}$, $S_{maj} = S_{def}$	**4D** $S_{min} = S_3$, $S_{maj} = S_{def}$	**4D** $S_{min} = S_{def}$, $S_{maj} = S_3$	**4D** $S_{min} = S_1$, $S_{maj} = S_3$	**4D** $S_{min} = S_3$, $S_{maj} = S_1$	**Optimal default**
abalone	67.94	0.55	-0.03	-0.07	-0.02	-0.05	-0.05	0.78
balance-scale	76.98	**-1.01**	**1.19**	**-3.65**	**1.19**	**-34.27**	**-8.85**	0.38
breast-cancer	64.98	-0.18	-0.13	-0.29	0.02	**-1.19**	**-4.95**	0.55
breast-w	97.53	-0.09	0.02	0.01	-0.01	0.03	0.01	0.09
car	96.74	-0.13	-0.48	0.20	0.36	**-1.85**	0.26	**-10.29**
cleveland	76.38	**1.37**	0.11	0.08	-0.05	-0.01	0.19	**3.79**
credit-g	69.90	-0.12	0.20	0.07	-0.05	-0.18	0.03	0.67
ecoli	88.82	0.89	0.64	-0.26	0.57	0.44	-0.07	**1.24**
glass	69.26	0.54	0.00	-0.44	0.00	0.56	-0.44	**4.89**
haberman	65.40	0.18	0.72	**-3.58**	**2.20**	-0.25	**-3.00**	0.70
hepatitis	79.00	0.82	-0.20	-0.20	0.00	**-2.20**	0.40	0.58
ionosphere	90.89	-0.18	0.00	0.00	0.00	0.00	0.00	0.59
mammography	89.70	-0.49	-0.39	-0.77	-0.35	-0.30	**-14.99**	0.38
new-thyroid	98.93	-0.44	0.00	0.00	0.00	-0.03	0.00	-0.33
nursery	99.90	-0.14	-0.06	-0.18	-0.01	-0.02	-0.18	-0.19
pima	72.87	-0.16	-0.03	-0.12	0.05	-0.03	-0.25	0.59
postoperative	43.66	**-2.11**	-0.69	**-2.81**	0.08	**2.35**	**-3.72**	0.68
transfusion	67.64	-0.70	**-1.33**	**-1.33**	0.00	**-9.94**	**-5.78**	**1.01**
vehicle	95.10	-0.21	0.00	0.00	0.00	0.00	0.00	0.38
yeast	84.95	-0.38	0.00	-0.64	-0.08	-0.08	-0.64	**1.98**
Averages of differences:		**-0.10**	**-0.02**	**-0.70**	**0.20**	**-2.35**	**-2.10**	**0.42**

Optimisation by the stratified cross-validation

Leave-one-out method of optimisation has some disadvantages (see e.g. [120] and the literature cited there). Thus, we tried to examine whether a change of the optimisation method in RIONIDA may lead to a difference in results. We have used the cross-validation method (more specifically, the 10-fold stratified cross-validation[15]) instead of leave-one-out in the optimisation during learning. Results in Table 5.22 (column with 'CV opt.' implementation) show that for the implementation using considered cross-validation for optimisation, for 1 data set, one can observe improvement by more than 1%, and for 2 data sets (one of them is *postoperative*, the smallest data set out of the used ones), one can observe worsening by more than 1%. The average of differences is equal to -0.1%. Hence, we can (roughly) conclude that this method does not help to obtain significant improvement.

Optimisation in 4-dimensional parameter space

It has been (experimentally) proved that the idea of the scaled generalised local decision rule (see Section 4.2) is beneficial for the RIONIDA algorithm. It is an extension of rules used in the RIONA algorithm.

We investigated the following further extension of this idea. Instead of using one scaling parameter $s \in \{-0.1\} \cup [0, 1]$, two scaling parameters of the rule are used, i.e. $s_{min} \in \{-0.1\} \cup [0, 1]$ and $s_{maj} \in \{-0.1\} \cup [0, 1]$. The difference to the standard sg-rule is that we use either parameter s_{min} or s_{maj} depending on the decision of the training example constituting the rule: s_{min} for the minority class and s_{maj} for the majority class. Also, instead of using one set S of admissible values of the parameter s, we use two separate sets S_{min}, S_{maj} of admissible values of the parameters s_{min}, s_{maj}, respectively.

In the learning phase, instead of searching for the optimal values of three parameters, we search for the optimal values of four parameters, including s_{min} and s_{maj}. In this way, we search for the optimal values of parameters in a 4-dimensional space.

We implemented all these ideas and experimented with a few different settings for sets S_{min}, S_{maj}. By default, we use $S_{min} = S_{def}$, $S_{maj} = S_{def}$ (see e.g. Subsection 5.5.4). Also, we used in the settings the sets $S_3 = \{-0.1\}$, $S_1 = \{1.0\}$ used previously in Subsection 5.5.4.

The considered extension of RIONIDA allows us to use different scaling factors for the minority and majority classes. Therefore, we expected that such extended implementation could lead to the improvement of RIONIDA performance. Results in Table 5.22 (fourth column with '4D' implementation, and default settings) show that for the implementation using 4-dimensional optimisation and default settings, for 1 data set, one can observe improvement by more than 1%; and for 1 data sets – worsening by more than 1%. The average of differences is equal to -0.02%. Hence, contrary to our expectations, we can (roughly) conclude that this method (with default settings) does not bring significant improvement.

[15]This should not be confused with the 10-fold stratified cross-validation used for algorithm evaluation (see Subsection 5.1.2).

We also experimented with 4 other settings for this implementation. The case with $S_{min} = S_3$, $S_{maj} = S_{def}$ corresponds to the situation when for all training examples belonging to the minority class we do not check the consistency of formed rules (kNN like method of voting). For setting, $S_{min} = S_{def}$, $S_{maj} = S_3$, vice versa, i.e. for all training examples belonging to the majority class we do not check the consistency of formed rules. For setting $S_{min} = S_1$, $S_{maj} = S_3$, for training examples belonging to the minority class pure rules are used while for those belonging to the majority class kNN like method of voting is used. For setting $S_{min} = S_3$, $S_{maj} = S_1$, vice versa, i.e. for the majority class pure rules are used while for the minority class kNN like method of voting is used.

For two last of these cases (see results in columns 7 and 8 in Table 5.22), one can observe significant worsening of the performance of RIONIDA. For both of these settings, for 5 or 6 data sets, one can observe worsening by more than 1% (sometimes even much more) and the average of differences below -2%. It shows that using different approaches (by means of either rule-based or instance-based) for the minority and majority classes is not beneficial.

For the case with $S_{min} = S_3$, $S_{maj} = S_{def}$, one can observe significant worsening: for 4 data sets, worsening by more than 1% and the average of differences is equal to -0.7%. Hence, we can (roughly) conclude that this method does not bring significant improvement. It seems incomprehensible since such setting, in a sense, allows for more frequent voting for the minority class. However, on the other hand, it shows that the balance between voting for the minority and majority classes should be acquired.

For the case with $S_{min} = S_{def}$, $S_{maj} = S_3$, the 'opposite' case to the previously discussed, one can observe small improvement. For two data sets, one can observe an improvement by more than 1%. The average of differences is equal to 0.2%. It is also worth mentioning that for all but 3 data sets absolute change is less than 0.5% (for 15 data sets less than 0.1%; a few times even equal to 0%). It should be noted that the two data sets for which a significant improvement was observed are: *balance-scale*, *haberman*; these are data sets for which using the full scale of admissible values ($S = S_{def}$) was beneficial (see e.g. comparison to the case $S = S_3$ in Table 5.20 with significantly reduced the scale; see also the considerations about *haberman* data set in Subsection 4.3.5). The considered setting and the obtained results show that it could be beneficial for some data sets to take into account the considered 4-dimensional extension of RIONIDA.

Optimisation of parameters with larger training data sets

Here, we consider the specific usage of RIONIDA, which is not typical for learning algorithm. This is due to the fact that we also make use of the information from the test data. This approach is somehow analogous (but not the same) to the methodology for the *max* strategy used for other algorithms than RIONIDA in the previous section.

Generally, the RIONIDA algorithm uses the given training set in two aspects: (1) for the optimisation of the values of internal parameters, and (2) as the base of examples for searching the closest cases to the considered test case. Here, we try to answer the question: Can we significantly improve the performance of RIONIDA by

providing more examples to the first-mentioned aspect of RIONIDA?

We try to answer this question by presenting the specific usage of RIONIDA, which learns the optimal values of parameters on the whole data set. Then it uses these learned values of parameters (k, p, s) as fixed in all runs of RIONIDA. During performance estimation of such classifier, it uses as usually the given training data, but with fixed values (as described above) of internal parameters.

Intuitively, one could expect that by using fixed values of parameters learned on the whole available data, all the changes in performance would be positive or at least very small in case of negative changes.

However, results in Table 5.22 (column with Optimal implementation) show that it is not valid in one case: for *car* data set, the worsening is greater than 10%. This case shows that using information about the optimal values of parameters for larger training data without using the data as the base of cases may be useless.

On the other hand, as expected, for all but 3 data sets the improvement can be observed (however only for 5 data sets higher than 1%); in other (2) cases (besides of *car* data set) the worsening is very slight (by less than 0.5%). If we examine these 5 data sets with improvement greater than 1%, it turns out that these are small data sets: 4 of them less than 1000 examples, and one less than 1500 examples. Moreover, the highest improvement (around 5% and 4%) was achieved for fairly small data sets (*glass* with 214 examples and *cleveland* with 303 examples).

Generally, we can (roughly) conclude that it is not necessary to possess a large number of examples in order to learn the internal parameters properly. It was recognised that increasing the number of training examples is especially important for small data sets. This observation is a promising fact for attempting to analyse big data sets (see Section 6.2, and also Subsection 5.4.3).

5.6 General summary of the described experiments

The most important experiments were presented in Section 5.3. These experiments were thoroughly designed in many aspects, taking into account that we deal with imbalanced learning problem (see Sections 5.1 and 5.2). In particular, we employed important steps in the process of evaluation of algorithms (see Sections 2.6 and 5.1); we selected for comparison two different performance measures, diverse imbalanced data sets, and various state-of-the-art algorithms; moreover, we also took into account that these algorithms can be used with different settings and preceded by different preprocessing filters. The final general conclusion is that for both used performance measures, regardless of whether we use default settings of algorithms or adjusted settings or even (potentially) learned settings by the meta-learning scheme, RIONIDA significantly outperforms any of the algorithms used in experiments (with the single exception pointed out in Subsection 5.3.3).

To understand more deeply the mentioned above exceptional results, we analysed the performance of RIONIDA during the performed experiments from the point of view of some additional aspects (see Section 5.4). It was shown that any of the internal parameters of RIONIDA (i.e. k, p, s) can boost its performance (see Subsection 5.4.1 and Chapter 4). The RIONIDA algorithm can learn the relevant values of these parameters precisely (and so proves to be highly effective). In Subsection 5.4.2, we

presented an example, illustrating that RIONIDA can deal with data sets containing many outliers. Moreover, we showed that using the scaled generalised local decision rules (and related to them learning the internal parameter s of RIONIDA) is crucial for such performance. Then, Subsection 5.4.3 showed that in many cases, the learned internal parameters are stable. This additionally proves their importance for RIONIDA. Moreover, it was experimentally shown for G-mean that for many data sets the learned optimal parameter p is close to the percentage of the minority class, what is consistent with the formulated and proved in the thesis Theorem 4.1. In Subsection 5.4.4, we also analysed the real-time of running of RIONIDA (training and testing). We found that it is comparable to other algorithms. Moreover, the presented observations suggest that the real-time of training and testing for RIONIDA is significantly less than given by the theoretical bounds (due to using indexing trees).

Section 5.5 presented the results from some additional experiments, which enabled us to investigate the RIONIDA algorithm in two aspects.

The first aspect was to understand even better the mentioned above exceptional results of RIONIDA. In Subsection 5.5.1, we (roughly) showed that RIONIDA realises the power of filters dedicated to imbalanced data and does much more for the relevant classification of imbalanced data. Then, in Subsection 5.5.2, we more deeply compared RIONIDA with its predecessor, RIONA (for G-mean as an example). We made a few observations, the most important of which are: (1) RIONIDA much better balances components of the considered measure (Sensitivity and Specificity in case of G-mean) than RIONA, (2) RIONIDA outperforms RIONA regardless of the used filter for RIONA, (3) for some data sets RIONIDA outperforms RIONA not only on G-mean but also on its both components, i.e. Sensitivity and Specificity, (4) using other settings for RIONA does not change the general comparison of these algorithms. All these observations confirm that the performance of RIONIDA cannot be obtained by using RIONA with proper filters, settings or even by better balancing Sensitivity and Specificity. RIONIDA is a more compound algorithm, and its results cannot be obtained by simple modifications of RIONA use. In Subsection 5.5.3, we also compared RIONIDA with an exemplary algorithm dedicated to imbalanced data, namely BRACID (the second best algorithm used in comparisons for the *defG* and *defF* strategies). RIONIDA outperforms BRACID (on both G-mean and F-measure) due to three general reasons: (1) RIONIDA better balances the primary components of the performance measure, (2) sometimes it outperforms BRACID on both primary components (Sensitivity and Specificity; or Sensitivity and Precision), (3) in case RIONIDA outperforms BRACID only on one of the components, the gain on this component is higher than the loss on the second one.

The second aspect was related to answering the question of whether RIONIDA performance can be further improved (for G-mean as an example). In Subsection 5.5.4, we used settings specific to RIONIDA. In Subsection 5.5.5, we used settings adopted from RIONA. The general conclusion is that none of these different settings leads to the improvement of the G-mean performance measure (in the context of the used data sets). However, considering the second part of the mentioned experiments, it should be noted that many of the methods designed with RIONA are oriented on optimisation of the Accuracy measure. They would have to be redesigned and reimplemented for measures specific to imbalanced data.

In particular, the attribute weighting method called *Perceptron* seems worthy to check in future experiments. Moreover, in Subsection 5.5.6, we used (additionally implemented) modified versions of RIONIDA. Using stratified cross-validation instead of the leave-one-out during optimisation does not lead to the improvement of the RIONIDA performance. However, the idea of two separate parameters for the scaled generalised local decision rules for the majority and minority classes (and in consequence, 4-dimensional optimisation) can lead to the improvement of RIONIDA performance. We have also shown that a relatively small number of examples is usually sufficient to learn the values of internal parameters of high quality.

To sum up, RIONIDA achieves impressively good results in comparison to the quality of the other state-of-the-art algorithms analysed in the thesis. All aspects of RIONIDA, i.e. using neighbours as in RIONA, using weights for two classes, and using the idea of scaled generalised local decision rule are essential for its performance. At the same time, RIONIDA running time is comparable to the used algorithms. It was far more successful to construct the RIONIDA algorithm than to use the RIONA algorithm with filters for imbalanced data or with different settings. Also, RIONIDA outperforms algorithms dedicated to imbalanced data not only on the chosen performance measures but also on the primary components of such measures (i.e. Sensitivity, Specificity, Precision). It is hard to improve the performance of RIONIDA with a simple change of settings. However, we proposed an extension of RIONIDA, which seems to be promising for future investigation.

Chapter 6

Final conclusions

The main goal of the thesis was to develop: (i) new methods based on combination of instance- and rule-based approaches, and (ii) systems based on these methods with a high quality of classification for different types of data sets. The realisation of this aim was divided into two steps: for balanced and imbalanced data.

6.1 Summary

In Chapter 2, we presented, known from the literature, the equivalence of specific lazy rule learning, for symbolic attributes only with the simple rule-based approach. This result was generalised in the next chapter (related to the RIONA algorithm) for more general rules commonly used in the thesis.

RIONA brings together some ideas of instance-based learning and rule induction into a single algorithm. It uses rules that group values for both numerical and symbolic attributes. RIONA is a lazy learning approach using only the rules on the basis of a neighbourhood of the test case. We (empirically) found that for correct classification of a test example generally, it is enough to consider only its small neighbourhood instead of the whole training set. This allowed us to develop an efficient algorithm without loss in Accuracy compared to the pure rule-based classifier. Also, we found that the appropriate selection of the neighbourhood size is a crucial factor for obtaining high Accuracy. We designed a method for efficient learning of the optimal size of the neighbourhood.

RIONA obtained the Accuracy comparable to the well-known systems. Theoretical results explain the relationships of the RIONA algorithm with both instance- and rule-based classifiers. On the basis of these results, we proposed a user-friendly explanation method of the decisions returned by the classifiers obtained from RIONA.

RIONIDA is an extension of RIONA combining the instance- and rule-based approaches for imbalanced data. Additionally, RIONIDA combines these approaches in another aspect, namely by using special rules, that are more general than the ones used in RIONA. This algorithm optimises the explicitly given performance measure. All main ideas embedded in RIONIDA are essential for obtaining the high quality of its performance including: optimisation of the fixed performance measure as well as three proposed internal parameters. This algorithm is relatively fast (both in

the training and testing phase). Moreover, the theoretical results concerning the parameter responsible for assigning relevant weights for the minority and majority classes can be used for acceleration of the training phase. The RIONIDA algorithm (as RIONA) has the desired property of explainability.

RIONIDA achieves impressively good results in comparison to the quality of the other state-of-the-art algorithms analysed in the thesis. RIONIDA significantly outperforms these algorithms on the chosen performance measures and systematically on their primary components (such as Sensitivity, Specificity, Precision) for selected algorithms. Moreover, running time of RIONIDA is comparable with those of the used algorithms. It was far more successful to construct the RIONIDA algorithm than to use the RIONA algorithm with filters for imbalanced data or different settings. It is hard to improve the performance of RIONIDA with a simple change of settings. However, we proposed an extension of RIONIDA, which seems to be worthy of future research.

In summarising, the claim about the realisation of the aim of the thesis mentioned above is justified.

6.2 Future works

There are several possible directions for future research related to: (1) extensions of RIONIDA, (2) continuation of presented experiments, (3) applying RIONIDA for more complex tasks. We present these issues in the order roughly from the easiest to the most challenging ones.

Technical things to do

In the future, the following useful implementations could be done:

- accelerated version of the learning phase of RIONIDA (see Subsection 4.5.3);
- special data structures in RIONIDA for identical objects to avoid slowing down computations in specific situations (see Subsection 5.4.4);
- optimisation with other performance measures not based on confusion matrix, in particular, AUC measure;
- inclusion of RIONIDA into WEKA.

Further experimental things to do

Further experiments with more in-depth analysis could be performed.

- Some of the additional experiments were performed only for G-mean performance measure (see Subsections 5.5.4 and 5.5.5). These experiments could be repeated using F-measure.
- More extensive experiments could be performed for RIONIDA (and different performance measures for imbalanced data) to more thoroughly

(experimentally) check the pre-assumed hypothesis stating that (1) there is no need to use the whole training set in the process of classification, and (2) the bound of the neighbourhood size can even improve the classification performance or at least not reduce it significantly (see Subsection 4.4.2 and also Subsubsection 'Different maximal k value' on page 208).

Further investigation on RIONIDA and its extensions

Although we investigated in a few directions whether RIONIDA can be further improved, much extensive investigation remains to be done. Also, the theoretical and practical properties of RIONIDA can be further explored. In particular, the following directions can be continued.

- We suppose that by using different voting methods jointly with appropriate set P (for admissible values of the parameter p) one could obtain improvement in the performance of RIONIDA (see 'Different voting methods' in Subsection 5.5.5). The relevant investigation with accompanying experiments could be performed to verify this hypothesis. This also applies to weighting method especially for *Perceptron* (see 'Different attribute weighting methods' in Subsection 5.5.5).

- The preliminary experiments show that using different parameter s for the majority and minority classes (and in consequence, 4-dimensional optimisation) can improve the performance of RIONIDA. More comprehensive investigation of this RIONIDA extension could be done.

- For different borderline regions, different optimal values of the parameter p (and possibly also for two other parameters) could be searched (see Subsection 4.3.3).

- Further effort can be made in the direction of Explainability of classifiers generated by RIONIDA.

- Theoretical results indicating the optimal value for the importance of the minority class can be extended (mathematically calculated) to more general circumstances occurring in practice. Current and future theoretical results could be used for faster searching for the actual optimal value of the parameter p.

Using RIONIDA for balanced data

Although RIONIDA was developed for imbalanced data, it can be easily used for balanced data with Accuracy as the performance measure. We have performed preliminary experiments in this aspect and noticed that for some data sets the obtained optimal parameter p was not equal to 0.5. Such setting can potentially help to construct a classifier with higher Accuracy than with the default value 0.5 for balanced data (as it is used for RIONA). Also, the parameter s can be used to find its optimal value for balanced data. This shows the possibility to use RIONIDA for balanced data with potentially better performance than RIONA. However, this issue should be further investigated.

Using RIONIDA for big data

In Section 5.4, we showed that for many data sets, some internal parameters of RIONIDA are stable. Also, we noticed in the same section that if for any internal parameter of RIONIDA its optimal values are stable in a part of the data set, then it can be an argument to use the same optimal values of the parameter for the larger part of the data set. We believe that these observations can be used to incrementally learn the optimal values of internal parameters, even on big training sets. Also, preliminary experiments showed that it is sufficient to use a relatively small amount of training objects to learn the internal parameters of RIONIDA with high quality (see Subsubsection 'Optimisation of parameters with larger training data sets' on page 216). These facts suggest that the use of RIONIDA for imbalanced big data can also be achievable and successful. However, extensive experiments for practical application of RIONIDA in such case should be done. In particular, investigation on the time complexity of original RIONIDA (using indexing trees) and its possible modifications for imbalanced big data sets should be done.

Other extensions of RIONIDA

Analysis of imbalanced data encompasses many issues which are covered in the thesis only marginally or not covered at all (see Section 1.3). All these issues open a wide field of investigations related to adjusting the model used currently for RIONIDA to be used successfully for real-life problems. These issues can be investigated in future works.

Appendices

Appendix A

Counter example for specific form of general rules

Let us try to redefine *GenRules* from Definition 2.8 (on page 34) to make the admissible conditions independent from the test example. Let us redefine the set *GenRules* in such a way that for symbolic attributes $a \in A$ the only admissible conditions are: $a \in B(v, \varrho_a(v, w))$, where $v, w \in V_a$. We call such redefined set as *GeneralRules*. Note that in the above mentioned definition v is independent from the test example in contrast to definition of *GenRules* given in the Definition 2.8. We construct from it the set of all maximally general rules $MaxRules(GeneralRules, trnSet)$ (see Definition 2.11). Analogously to Theorems 2.2, 3.2, 3.4 one could formulate the following hypothetical theorem:

Theorem A.1. *The rule g-rule* $\left(tst, trn, \{\varrho_a\}_{a \in A_{sym}}\right)$ *for the test object tst and any training object* $trn \in trnSet$ *is consistent with the training set trnSet if and only if there exists rule* $r \in MaxRules(GeneralRules, trnSet)$ *covering examples tst and trn.*

However this theorem is not true. To show it, we construct the relevant counterexample.

Suppose that we have the traning set, *trnSet* and the test example, *tst* as given in Table A.1. It should be noted that the values of the conditional attribute *BloodGroup* and the decision attribute *Diagnosis* are the same in Table 2.1. Thus, the distances between values of the attribute *BloodGroup* are the same as calculated in Subsection 2.2.2 and graphically shown in Figure 2.1.

First, let us construct for the test object *tst* and the training object trn_7, the *g-rule* (tst, trn_7) which is equal to

$$\text{if } Gender = F \wedge \ BG \in B(AB, \varrho_{BG}(AB, 0)) \text{ then } Diagnosis = Healthy.$$

Taking into account the distances calculated in Subsection 2.2.2 this is equivalent to

$$\text{if } Gender = F \wedge \ BG \in \{AB, B, 0\} \text{ then } Diagnosis = Healthy.$$

This rule is inconsistent with *trnSet* because example trn_5 satisfies it and has different decision than the considered rule.

Object	***Gender***	***BloodGroup* (*BG*)**	***Diagnosis***
trn_1	M	A	Sick
trn_2	M	AB	Sick
trn_3	F	AB	Healthy
trn_4	F	AB	Healthy
trn_5	F	B	Sick
trn_6	M	B	Healthy
trn_7	F	0	Healthy
tst	F	AB	?

Table A.1: Artificial data set with 2 conditional attributes (*Gender* and *BloodGroup*, in short, BG; both symbolic) and decision attribute *Diagnosis*. Seven objects derive from training set and the last object is a test object (its decision is unknown).

Let us take in the above definition of *GeneralRules* the admissible condition for attribute BG with $v = 0$ and $w = AB$. This condition is as follows:

$$BG \in B(0, \varrho_{BG}(0, AB)).$$

Taking into account the distances calculated in Subsection 2.2.2 this condition is equivalent to the following one:

$$BG \in \{0, AB\}.$$

Then, the rule

$$\text{if } Gender = F \wedge \ BG \in \{0, AB\} \text{ then } Diagnosis = Healthy$$

belongs to the set $MaxRules(GeneralRules, trnSet)$, because: it belongs to the set *GeneralRules*; it is consistent and maximal (e.g. for attribute BG the more general condition than $BG \in \{0, AB\}$ is the trivial condition, which would produce an inconsistent rule). Moreover, this rule covers both examples tst and trn_7. Hence, for such definition of the set *GeneralRules* Theorem A.1 does not hold.

Appendix B

An example of the macro- or micro-averaging of results of cross-validation

It was experimentally shown in [67] that the simple averaging of partial results of cross-validation, i.e. the macro-average style can give a biased estimation. We wanted also to check whether using the macro- or micro-average (see Subsection 2.6.2) can give a different final result of comparison of a learning algorithm (with another one) in terms of better or worse. The latter means a sign of the differences of averaged (macro or micro) partial results (of two algorithms) of a given performance measure. Below, we show that the answer is positive and we give a clear example for it. To show it we chose G-mean as the performance measure.

Even using 10 times repeated 10-fold cross-validation (more precisely stratified cross-validation) we found such data set[1] and such pair of algorithms (let us call them A and B) for which such differences occur. It means that if we use the macro-average, then algorithm A turns out to be better than B, and if we use the micro-average, then vice versa, i.e. B turns out to be better than A. To show the effect of different choices of averaging we selected one group of results out of ten cross-validations which shows this effect the most.

In one of the 10-fold cross-validation, the differences between algorithms A and B were 2.52% and −1.13%, respectively. From this real example, we have chosen three partial results (from three splits) and in this way we constructed a possible result for the 3-fold cross-validation for two learning algorithms. We tried to make the example as simple as possible.

In Table B.1 we show such an example of potential results of running learning algorithms A and B for 3-fold cross-validation (in fact stratified cross-validation). This table for each fold of cross-validation presents: number of a fold (cv), numbers from confusion matrix i.e. TP, FN, FP, TN (see Subsection 2.6.1), and G-mean calculated for the given fold. After results for three folds, average numbers of these three G-mean values are given for both algorithms (for the algorithm A it is 34.82% and for the algorithm B it is 28.93%). This is the effect of the macro-average computing for G-mean.

[1] In fact, we found two such data sets.

Below the third horizontal line are given sums of coefficients of the joint confusion matrix (with coefficients equal to the sum of coefficients from each fold). For such joint matrix, G-mean measures are given for both algorithms (for the algorithm A it is 36.27% and for the algorithm B it is 42.15%). This is the effect of the micro-average computing for G-mean.

In the last column, differences between algorithms are given depending on whether the macro- or micro-average is used. It means that the algorithm A is better than the algorithm B by around 6% if the macro-average is used. On the other hand, the algorithm A is worse than the algorithm B by around 6% if the micro-average is used. It means that depending on the selection of the macro- or micro-averaging completely different conclusions can be drawn (and these differences are similar and relatively high). This artificial example (yet constructed on the base of real experiments) shows that one has to be very careful with the way of averaging of the partial results of cross-validation.

According to [67] the micro-average style should be used for F-measure since it gives less bias. We expect that analogously for G-mean the micro-average gives less bias. Thus, we expect that for G-mean also the micro-average style should be used.

Thus, in the experiments shown in the thesis, the micro-average style of computation was used for all performance measures, i.e. F-measure and G-mean.

Table B.1: An example of the macro- or micro-averaging of results of cross-validation. Line 6 of the table presents the results of the macro-averaging and the last line presents results of the micro-averaging.

	results for learning alg. A					**results for learning alg. B**					diff-
cv	TP	FN	FP	TN	G-mean	TP	FN	FP	TN	G-mean	errence
1	2	1	2	4	66.67%	0	3	1	5	0.00%	
2	0	3	6	0	0.00%	1	2	4	2	33.33%	
3	2	0	6	1	37.80%	2	0	5	2	53.45%	
	macro-average G-mean				**34.82%**					**28.93%**	**5.89%**
$\sum$	*TP*	*FN*	*FP*	*TN*	G-mean	*TP*	*FN*	*FP*	*TN*	G-mean	
	4	4	14	5	**36.27%**	3	5	10	9	**42.15%**	**-5.87%**

Appendix C

Remark on the localisation of the optimal parameter p

Fact C.1. *Under the assumptions of Theorem 4.1 we have*

$$\inf_{p_{opt}}|p_{opt}-q| \leq \frac{1}{k}\cdot\max\begin{cases}\sup\limits_{k',q:\mu<M,F(M)+F(M-1)<1}(M-\mu)\\ \sup\limits_{k',q:\mu>M,F(M)+F(M-1)>1}(\mu-M)\,,\end{cases}$$

where $F=F_{B(k',q)}$, $M=M(k',q)$ and $\mu=\mu(k',q)=k'q$ are respectively the median and the mean of $B(k',q)$.

Proof. We use the proof of the previous theorem.

We consider three cases. First, if $F(M)-\frac{1}{2}=\frac{1}{2}-F(M-1)$, then the interval of the optimal values p_{opt} is equal to $[\tilde{p}_{opt},\tilde{p}_{opt}+\frac{2}{k})$. Then q is within this interval.

Second, if $F(M)-\frac{1}{2}<\frac{1}{2}-F(M-1)$, then $\tilde{p}_{opt}$ is such that $M=\tilde{p}_{opt}k$ and the interval of the optimal values p_{opt} is equal to $[\tilde{p}_{opt},\tilde{p}_{opt}+\frac{1}{k})$. If $q>\tilde{p}_{opt}$, then from first part of Equation 4.8 we have that q is within this interval (and the distance is zero). Thus, in this case we have

$$\inf_{p_{opt}}|p_{opt}-q| \leq \sup_{q<\tilde{p}_{opt}}(\tilde{p}_{opt}-q)=\frac{1}{k}\cdot\sup_{kq<k\tilde{p}_{opt}}(k\tilde{p}_{opt}-kq)=\frac{1}{k}\cdot\sup_{\mu<M}(M-\mu)\,,$$

where $\sup\limits_{\mu<M}(M-\mu)$ is $\sup\limits_{k',q:\mu(k',q)<M(k',q)}(M(k',q)-\mu(k',q))$ and analogously in similar places.

Third, if $F(M)-\frac{1}{2}>\frac{1}{2}-F(M-1)$, then $\tilde{p}_{opt}$ is such that $M=\tilde{p}_{opt}k+1$ and the interval of the optimal values p_{opt} is equal to $[\tilde{p}_{opt},\tilde{p}_{opt}+\frac{1}{k})$. If $q<\tilde{p}_{opt}+\frac{1}{k}$, then from the second part of Equation 4.8, we have that q is within this interval (and the distance is zero). Thus, in this case we have

$$\begin{aligned}\inf_{p_{opt}}|p_{opt}-q| &\leq \sup_{q>\tilde{p}_{opt}+\frac{1}{k}}\left(q-\left(\tilde{p}_{opt}+\frac{1}{k}\right)\right)=\frac{1}{k}\cdot\sup_{kq>k\tilde{p}_{opt}+1}(kq-(k\tilde{p}_{opt}+1))\\ &=\frac{1}{k}\cdot\sup_{\mu>M}(\mu-M))\,.\end{aligned}$$

Hence,

$$\inf_{p_{opt}} |p_{opt} - q| \leq \frac{1}{k} \cdot \max \begin{cases} \sup\limits_{\mu<M} (M - \mu) & \text{when F(M) + F(M-1)} < 1 \\ \sup\limits_{\mu>M} (\mu - M) & \text{when F(M) + F(M-1)} > 1. \end{cases}$$

□

It seems[1] that assuming that $\mu < \frac{1}{2}$, one can find a much better bound for the max value from the above fact than $\ln 2$ (independently of q and k). In particular, we expect that the first sup from the above fact is equal to 0.

[1] This was verified for some selected values of n. Taking into account the performed experiments (see Chapter 5), the more general intuition for $n = 1, 2, \ldots, 100$ would be more relevant. This, not very complicated task, was left for the future work.

Bibliography

[1] Rseslib 3: Rough set and machine learning open source in Java. http://rseslib.mimuw.edu.pl.

[2] Weka 3: Machine Learning Software in Java. https://www.cms.waikato.ac.nz/ml/weka/.

[3] Amina Adadi and Mohammed Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018. doi:10.1109/ACCESS.2018.2870052.

[4] Charu C. Aggarwal. *Outlier Analysis*. Springer, New York, NY, 1st edition, 2013. doi:10.1007/978-1-4614-6396-2.

[5] Charu C. Aggarwal. Instance-Based Learning: A Survey. In Charu C. Aggarwal (ed.), *Data Classification: Algorithms and Applications*, pp. 157–186. Chapman & Hall/CRC, New York, 1st edition, 2014. doi:10.1201/b17320.

[6] David W. Aha (ed.). *Lazy Learing*. Springer, Dordrecht, 1st edition, 1997. doi:10.1007/978-94-017-2053-3.

[7] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991. doi:10.1023/A:1022689900470.

[8] Ammar Almasri, Erbug Celebi, and Rami S. Alkhawaldeh. EMT: Ensemble Meta-Based Tree Model for Predicting Student Performance. *Scientific Programming*, 2019:Article No. 3610248, 1–13, 2019. doi:10.1155/2019/3610248.

[9] Aijun An, Nick Cercone, and Xiangji Huang. A Case Study for Learning from Imbalanced Data Sets. In *Advances in Artificial Intelligence (Canadian AI 2001)*, pp. 1–15. Springer, Heidelberg, 2001. doi:10.1007/3-540-45153-6_1.

[10] Martin H. G. Anthony and Norman Biggs. *Computational Learning Theory: An Introduction*. Cambridge University Press, Cambridge, 1992.

[11] Muhammad Nafees Anwar. *Complexity measurement for dealing with class imbalance problems in classification modelling*. PhD thesis, Massey University, 2012.

[12] Sanghamitra Bandyopadhyay and Sriparna Saha. *Unsupervised Classification: Similarity Measures, Classical and Metaheuristic Approaches, and Applications*. Springer-Verlag, Heidelberg, 1st edition, 2013. doi:10.1007/978-3-642-32451-2.

[13] Sukarna Barua, Md. Monirul Islam, Xin Yao, and Kazuyuki Murase. MWMOTE–Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425, 2014. doi:10.1109/TKDE.2012.232.

[14] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, 2004. doi:10.1145/1007730.1007735.

[15] Jan G. Bazan. Discovery of Decision Rules by Matching New Objects Against Data Tables. In *Rough Sets and Current Trends in Computing (RSCTC 1998)*, pp. 521–528. Springer, Heidelberg, 1998. doi:10.1007/3-540-69115-4_72.

[16] Jan G. Bazan and Marcin Szczuka. RSES and RSESlib – A Collection of Tools for Rough Set Computations. In *Rough Sets and Current Trends in Computing (RSCTC 2001)*, pp. 106–113. Springer, Heidelberg, 2001. doi:10.1007/3-540-45554-X_12.

[17] Jan G. Bazan, Hung Son Nguyen, Sinh Hoa Nguyen, Piotr Synak, and Jakub Wróblewski. Rough Set Algorithms in Classification Problem. In Polkowski et al. [167], pp. 49–88. doi:10.1007/978-3-7908-1840-6_3.

[18] Mohamed Bekkar, Hassiba Kheliouane Djemaa, and Taklit Akrouf Alitouche. Evaluation measures for models assessment over imbalanced data sets. *Journal of Information Engineering and Applications*, 3(10):27–38, 2013.

[19] Aurélien Bellet, Amaury Habrard, and Marc Sebban. *Metric Learning*. Morgan & Claypool Publishers, San Rafael, CA, 2015. doi:10.2200/S00626ED1V01Y201501AIM030.

[20] Colin Bellinger. *Beyond the Boundaries of SMOTE: A Framework for Synthetically Oversampling the Manifold*. PhD thesis, University of Ottawa, 2016.

[21] Colin Bellinger, Christopher Drummond, and Nathalie Japkowicz. Manifold-based synthetic oversampling with manifold conformance estimation. *Machine Learning*, 107(3):605–637, 2018. doi:10.1007/s10994-017-5670-4.

[22] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. *Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[23] Cigdem Beyan and Robert Fisher. Classifying imbalanced data sets using similarity based hierarchical decomposition. *Pattern Recognition*, 48(5):1653–1672, 2015. doi:10.1016/j.patcog.2014.10.032.

[24] Yoram Biberman. A context similarity measure. In *Proceedings of the 7th European Conference on Machine Learning (ECML 1994)*, pp. 49–63. Springer, Heidelberg, 1994. doi:10.1007/3-540-57868-4_50.

[25] Jerzy Błaszczyński, Magdalena Deckert, Jerzy Stefanowski, and Szymon Wilk. Integrating Selective Pre-processing of Imbalanced Data with Ivotes Ensemble. In *Rough Sets and Current Trends in Computing (RSCTC 2010)*, pp. 148–157. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-13529-3_17.

[26] Jerzy Błaszczyński, Salvatore Greco, and Roman Słowiński. Inductive discovery of laws using monotonic rules. *Engineering Applications of Artificial Intelligence*, 25(2):284–294, 2012. doi:10.1016/j.engappai.2011.09.003.

[27] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. *Feature Selection for High-Dimensional Data*. Springer, Cham, 1st edition, 2015. doi:10.1007/978-3-319-21858-8.

[28] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity Measures for Categorical Data: A Comparative Evaluation. In *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM)*, pp. 243–254. SIAM, 2008. doi:10.1137/1.9781611972788.22.

[29] Remco R. Bouckaert and Eibe Frank. Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2004)*, pp. 3–12. Springer, Heidelberg, 2004. doi:10.1007/978-3-540-24775-3_3.

[30] Jeffrey P. Bradford, Clayton Kunz, Ron Kohavi, Cliff Brunk, and Carla E. Brodley. Pruning decision trees with misclassification costs. In *Proceedings of the 10th European Conference on Machine Learning (ECML 1998)*, pp. 131–136. Springer, Heidelberg, 1998. doi:10.1007/BFb0026682.

[31] Paula Branco, Luís Torgo, and Rita P. Ribeiro. A Survey of Predictive Modeling on Imbalanced Domains. *ACM Computing Surveys*, 49(2):Article No. 31, 1–50, 2016. doi:10.1145/2907070.

[32] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. doi:10.1023/A:1010933404324.

[33] Arlen Brown and Carl Pearcy. *An Introduction to Analysis*. Springer-Verlag, New York, NY, 1st edition, 1995. doi:10.1007/978-1-4612-0787-0.

[34] Borja Calvo and Guzmán Santafé. scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems. *The R Journal*, 8(1):248–256, 2016. doi:10.32614/RJ-2016-017.

[35] Alain Célisse and Tristan Mary-Huard. Exact Cross-Validation for kNN and application to passive and active learning in classification. *Journal de la Société Française de Statistique*, 152(3):83–97, 2011.

[36] Nitesh V. Chawla. Data Mining for Imbalanced Datasets: An Overview. In Oded Maimon and Lior Rokach (eds.), *Data Mining and Knowledge Discovery Handbook*, pp. 853–867. Springer, Boston, MA, 2005. doi:10.1007/0-387-25465-X_40.

[37] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and William P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16: 321–357, 2002. doi:10.1613/jair.953.

[38] Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kołcz. Editorial: Special Issue on Learning from Imbalanced Data Sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004. doi:10.1145/1007730.1007733.

[39] Debo Cheng, Shichao Zhang, Zhenyun Deng, Yonghua Zhu, and Ming Zong. kNN Algorithm with Data-Driven k Value. In *Advanced Data Mining and Applications*, pp. 499–512. Springer, Cham, 2014. doi:10.1007/978-3-319-14717-8_39.

[40] David A. Cieslak and Nitesh V. Chawla. Learning Decision Trees for Unbalanced Data. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2008)*, pp. 241–256. Springer, Heidelberg, 2008. doi:10.1007/978-3-540-87479-9_34.

[41] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine learning*, 3:261–283, 1989. doi:10.1007/BF00116835.

[42] William W. Cohen. Fast Effective Rule Induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML 1995)*, pp. 115–123. Morgan Kaufmann, San Francisco, CA, 1995. doi:10.1016/b978-1-55860-377-6.50023-2.

[43] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 3rd edition, 2009.

[44] Scott Cost and Steven Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57–78, 1993. doi:10.1007/BF00993481.

[45] Shounak Datta and Swagatam Das. Near-Bayesian Support Vector Machines for imbalanced data classification with equal or unequal misclassification costs. *Neural Networks*, 70:39–52, 2015. doi:10.1016/j.neunet.2015.06.005.

[46] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. An Exact Algorithm for F-Measure Maximization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS 2011)*, pp. 1404–1412. Curran Associates Inc., Red Hook, NY, 2011.

[47] Janez Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[48] Nilanjan Dey, Samarjeet Borah, Rosalina Babo, and Amira S. Ashour (eds.). *Social Network Analytics: Computational Research Methods and Techniques*. Academic Press, London, 1st edition, 2019. doi:10.1016/C2017-0-02844-6.

[49] Matías Di Martino, Federico Decia, Juan Molinelli, and Alicia Fernández. Improving electric fraud detection using class imbalance strategies. In *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods (ICPRAM 2012)*, volume 1, pp. 135–141. SciTePress, Setúbal, 2012. doi:10.5220/0003768401350141.

[50] Thomas G. Dietterich. Machine Learning Research: Four Current Directions. *AI Magazine*, 18:97–136, 1997.

[51] Thomas G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1923, 1998. doi:10.1162/089976698300017197.

[52] Shuya Ding, Bilal Mirza, Zhiping Lin, Jiuwen Cao, Xiaoping Lai, Tam V. Nguyen, and Jose Sepulveda. Kernel based online learning for imbalance multiclass classification. *Neurocomputing*, 277:139–148, 2018. doi:10.1016/j.neucom.2017.02.102.

[53] Pedro Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24 (2):141–168, 1996. doi:10.1007/BF00058656.

[54] Pedro Domingos and Michael Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29(2):103–130, 1997. doi:10.1023/A:1007413511361.

[55] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 0210–0215. Croatian Society MIPRO, 2018. doi:10.23919/MIPRO.2018.8400040.

[56] Harshit Dubey and Vikram Pudi. Class Based Weighted K-Nearest Neighbor over Imbalance Dataset. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2013)*, pp. 305–316. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-37456-2_26.

[57] Charles Elkan. The Foundations of Cost-Sensitive Learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pp. 973–978. Morgan Kaufmann, San Francisco, CA, 2001.

[58] Seyda Ertekin, Jian Huang, Léon Bottou, and Lee Giles. Learning on the Border: Active Learning in Imbalanced Data Classification. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007)*, pp. 127–136. ACM, New York, NY, 2007. doi:10.1145/1321440.1321461.

[59] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37–54, 1996. doi:10.1609/aimag.v17i3.1230.

[60] Alberto Fernández, Victoria López, Mikel Galar, María José del Jesus, and Francisco Herrera. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, 42:97–110, 2013. doi:10.1016/j.knosys.2013.01.018.

[61] Alberto Fernández, Sara del Río, Nitesh V. Chawla, and Francisco Herrera. An insight into imbalanced Big Data classification: outcomes and challenges. *Complex & Intelligent Systems*, 3(2):105–120, 2017. doi:10.1007/s40747-017-0037-9.

[62] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from Imbalanced Data Sets*. Springer, Cham, 1st edition, 2018. doi:10.1007/978-3-319-98074-4.

[63] Alberto Fernández, Salvador García, Francisco Herrera, and Nitesh V. Chawla. SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-Year Anniversary. *Journal of Artificial Intelligence Research*, 61(1):863–905, 2018. doi:10.1613/jair.1.11192.

[64] Helmut Finner. On a Monotonicity Problem in Step-Down Multiple Test Procedures. *Journal of the American Statistical Association*, 88(423):920–923, 1993. doi:10.2307/2290782.

[65] Peter A. Flach. The Geometry of ROC Space: Understanding Machine Learning Metrics through ROC Isometrics. In *Proceedings of the 20th International Conference on International Conference on Machine Learning (ICML 2003)*, pp. 194–201. AAAI Press, 2003.

[66] Peter A. Flach and Meelis Kull. Precision-Recall-Gain Curves: PR Analysis Done Right. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS 2015)*, pp. 838–846. MIT Press, Cambridge, MA, 2015.

[67] George Forman and Martin Scholz. Apples-to-Apples in Cross-Validation Studies: Pitfalls in Classifier Performance Measurement. *ACM SIGKDD Explorations Newsletter*, 12(1):49–57, 2010. doi:10.1145/1882471.1882479.

[68] Eibe Frank and Ian H. Witten. Generating Accurate Rule Sets Without Global Optimization. In *Proceedings of the 15th International Conference on Machine Learning (ICML 1998)*, pp. 144–151. Morgan Kaufmann, San Francisco, CA, 1998.

[69] Yoav Freund and Robert E. Schapire. Experiments with a New Boosting Algorithm. In *Proceedings of the 13th International Conference on Machine Learning (ICML 1996)*, pp. 148–156. Morgan Kaufmann, San Francisco, CA, 1996.

[70] Milton Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937. doi:10.1080/01621459.1937.10503522.

[71] Johannes Fürnkranz and Gerhard Widmer. Incremental Reduced Error Pruning. In *Proceedings of the 11th International Conference on Machine Learning (ICML 1994)*, pp. 70–77. Morgan Kaufmann, San Francisco, CA, 1994. doi:10.1016/B978-1-55860-335-6.50017-9.

[72] Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrac. *Foundations of Rule Learning*. Cognitive Technologies. Springer, Heidelberg, 2012. doi:10.1007/978-3-540-75197-7.

[73] Salvador García and Francisco Herrera. An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[74] Salvador García, Alberto Fernández, and Francisco Herrera. Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems. *Applied Soft Computing*, 9(4):1304–1314, 2009. doi:10.1016/j.asoc.2009.04.004.

[75] Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10): 2044–2064, 2010. doi:10.1016/j.ins.2009.12.010.

[76] Vicente García, Jose Sánchez, and Ramon Mollineda. An Empirical Study of the Behavior of Classifiers on Imbalanced and Overlapped Data Sets. In *Progress in Pattern Recognition, Image Analysis and Applications (CIARP 2007)*, pp. 397–406. Springer, Heidelberg, 2007. doi:10.1007/978-3-540-76725-1_42.

[77] Nicolás García-Pedrajas, Juan A. Romero del Castillo, and Gonzalo Cerruela-García. A Proposal for Local k Values for k-Nearest Neighbor Rule. *IEEE Transactions on Neural Networks and Learning Systems*, 28(2):470–475, 2017. doi:10.1109/TNNLS.2015.2506821.

[78] Anil K. Ghosh. On optimum choice of k in nearest neighbor classification. *Computational Statistics & Data Analysis*, 50(11):3113–3123, 2006. doi:10.1016/j.csda.2005.06.007.

[79] Anil K. Ghosh. On Nearest Neighbor Classification Using Adaptive Choice of k. *Journal of Computational and Graphical Statistics*, 16(2):482–502, 2007. doi:10.1198/106186007X208380.

[80] Jean D. Gibbons and John W. Pratt. P-Values: Interpretation and Methodology. *The American Statistician*, 29(1):20–25, 1975. doi:10.2307/2683674.

[81] Grzegorz Góra and Arkadiusz Wojna. RIONA: A New Classification System Combining Rule Induction and Instance-Based Learning. *Fundamenta Informaticae*, 51(4):369–390, 2002.

[82] Grzegorz Góra and Arkadiusz Wojna. RIONA: A Classifier Combining Rule Induction and K-nn Method with Automated Selection of Optimal Neighbourhood. In *Proceedings of the 13th European Conference on Machine Learning (ECML 2002)*, pp. 111–123. Springer-Verlag, Heidelberg, 2002. doi:10.1007/3-540-36755-1_10.

[83] Grzegorz Góra and Arkadiusz Wojna. Local Attribute Value Grouping for Lazy Rule Induction. In *Rough Sets and Current Trends in Computing (RSCTC 2002)*, pp. 405–412. Springer, Heidelberg, 2002. doi:10.1007/3-540-45813-1_53.

[84] Lacrimioara Grama and Corneliu Rusu. Choosing an accurate number of mel frequency cepstral coefficients for audio classification purpose. In *Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis (ISPA 2017)*, pp. 225–230, 2017. doi:10.1109/ISPA.2017.8073600.

[85] Lacrimioara Grama and Corneliu Rusu. Adding audio capabilities to TIAGo service robot. In *2018 International Symposium on Electronics and Telecommunications (ISETC)*, pp. 1–4, 2018. doi:10.1109/ISETC.2018.8583897.

[86] Jerzy W. Grzymala-Busse. LERS-A System for Learning from Examples Based on Rough Sets. In Roman Słowiński (ed.), *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, pp. 3–18. Springer, Dordrecht, 1992. doi:10.1007/978-94-015-7975-9_1.

[87] Jerzy W. Grzymala-Busse. Applications of the Rule Induction System LERS. In Polkowski and Skowron [166], pp. 366–375.

[88] Jerzy W. Grzymala-Busse and Witold J. Grzymala-Busse. Handling Missing Attribute Values. In Oded Maimon and Lior Rokach (eds.), *Data Mining and Knowledge Discovery Handbook*, pp. 37–57. Springer, Boston, MA, 2005. doi:10.1007/0-387-25465-X_3.

[89] Jerzy W. Grzymala-Busse, Linda K. Goodwin, Witold J. Grzymala-Busse, and Xinqun Zheng. An Approach to Imbalanced Data Sets Based on Changing Rule Strength. In Sankar K. Pal, Lech Polkowski, and Andrzej Skowron (eds.), *Rough-Neural Computing: Techniques for Computing with Words*, pp. 543–553. Springer, Heidelberg, 2004.

[90] Jerzy W. Grzymala-Busse, Jerzy Stefanowski, and Szymon Wilk. A Comparison of Two Approaches to Data Mining from Imbalanced Data. *Journal of Intelligent Manufacturing*, 16 (6):565–573, 2005. doi:10.1007/s10845-005-4362-2.

[91] Qiong Gu, Li Zhu, and Zhihua Cai. Evaluation Measures of the Classification Performance of Imbalanced Data Sets. In *Computational Intelligence and Intelligent Systems (ISICA 2009)*, pp. 461–471. Springer, Heidelberg, 2009. doi:10.1007/978-3-642-04962-0_53.

[92] Isabelle Guyon and André Elisseeff. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[93] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A. Zadeh (eds.). *Feature Extraction: Foundations and Applications*. Springer-Verlag, Heidelberg, 1st edition, 2006. doi:10.1007/978-3-540-35488-8.

[94] Hani Hagras. Toward Human-Understandable, Explainable AI. *Computer*, 51(9):28–36, 2018. doi:10.1109/MC.2018.3620965.

[95] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017. doi:10.1016/j.eswa.2016.12.035.

[96] Kais Hamza. The smallest uniform upper bound on the distance between the mean and the median of the binomial and Poisson distributions. *Statistics & Probability Letters*, 23(1):21–25, 1995. doi:10.1016/0167-7152(94)00090-U.

[97] David J. Hand. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning*, 77(1):103–123, 2009. doi:10.1007/s10994-009-5119-5.

[98] David J. Hand, Peter Christen, and Nishadi Kirielle. Z*: an interpretable transformation of the F-measure. *Machine Learning*, 110(3):451–456, 2021. doi:10.1007/s10994-021-05964-1.

[99] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, NY, 2nd edition, 2009. doi:10.1007/978-0-387-84858-7.

[100] Douglas M. Hawkins. *Identification of outliers*. Monographs on Applied Probability and Statistics. Springer, Dordrecht, 1980. doi:10.1007/978-94-015-3994-4.

[101] Haibo He and Edwardo A. Garcia. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. doi:10.1109/TKDE.2008.239.

[102] Haibo He and Yunqian Ma. *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley-IEEE Press, Piscataway, NJ, 1st edition, 2013.

[103] Robert C. Holte, Liane E. Acker, and Bruce W. Porter. Concept Learning and the Problem of Small Disjuncts. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI 1989)*, volume 1, pp. 813–818. Morgan Kaufmann, San Francisco, CA, 1989.

[104] Paul Hopkin. *Fundamentals of Risk Management: Understanding, evaluating and implementing effective risk management*. Kogan Page, London, 5th edition, 2018.

[105] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning Deep Representation for Imbalanced Classification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5375–5384, 2016. doi:10.1109/CVPR.2016.580.

[106] Mansoor Zolghadri Jahromi, Elham Parvinnia, and Robert John. A method of learning weighted similarity function to improve the performance of nearest neighbor. *Information Sciences*, 179(17):2964–2973, 2009. doi:10.1016/j.ins.2009.04.012.

[107] Andrzej Janusz. Algorithms for Similarity Relation Learning from High Dimensional Data. In James F. Peters and Andrzej Skowron (eds.), *Transactions on Rough Sets XVII*, pp. 174–292. Springer, Heidelberg, 2014. doi:10.1007/978-3-642-54756-0_7.

[108] Nathalie Japkowicz. The Class Imbalance Problem: Significance and Strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI 2000)*, pp. 111–117, 2000.

[109] Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, Cambridge, 2011. doi:10.1017/CBO9780511921803.

[110] Nathalie Japkowicz and Shaju Stephen. The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analysis*, 6(5):429–449, 2002. doi:10.3233/IDA-2002-6504.

[111] Nathalie Japkowicz, Catherine Myers, and Mark Gluck. A Novelty Detection Approach to Classification. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 1995)*, pp. 518–523. Morgan Kaufmann, San Francisco, CA, 1995.

[112] Liangxiao Jiang, Zhihua Cai, Dianhong Wang, and Siwei Jiang. Survey of Improving K-Nearest-Neighbor for Classification. In *4th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, pp. 679–683, 2007. doi:10.1109/FSKD.2007.552.

[113] Justin M. Johnson and Taghi M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6:Article No. 27, 1–54, 2019. doi:10.1186/s40537-019-0192-5.

[114] Mahesh V. Joshi, Ramesh C. Agarwal, and Vipin Kumar. Mining Needle in a Haystack: Classifying Rare Classes via Two-Phase Rule Induction. *ACM SIGMOD Record*, 30(2):91–102, 2001. doi:10.1145/376284.375673.

[115] Rob Kaas and Jan M. Buhrman. Mean, Median and Mode in Binomial Distributions. *Statistica Neerlandica*, 34(1):13–18, 1980. doi:10.1111/j.1467-9574.1980.tb00681.x.

[116] Harsurinder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. A Systematic Review on Imbalanced Data Challenges in Machine Learning: Applications and Solutions. *ACM Computing Surveys*, 52(4):1–36, 2019. doi:10.1145/3343440.

[117] Kittisak Kerdprasop and Nittaya Kerdprasop. A data mining approach to automate fault detection model development in the semiconductor manufacturing process. *International Journal of Mechanics*, 5(4):336–344, 2011.

[118] Mohammed Khalilia, Sounak Chakraborty, and Mihail Popescu. Predicting disease risks from highly imbalanced data using random forest. *BMC Medical Informatics and Decision Making*, 11(51):1–13, 2011. doi:10.1186/1472-6947-11-51.

[119] Claudia Klüppelberg, Daniel Straub, and Isabell M. Welpe. *Risk - A Multidisciplinary Introduction*. Springer, Cham, 2014. doi:10.1007/978-3-319-04486-6.

[120] Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 1995)*, volume 2, pp. 1137–1143. Morgan Kaufmann, San Francisco, CA, 1995.

[121] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016. doi:10.1007/s13748-016-0094-0.

[122] Bartosz Krawczyk, Michał Woźniak, and Gerald Schaefer. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, 14:554–562, 2014. doi:10.1016/j.asoc.2013.08.014.

[123] Evan Kriminger, José C. Príncipe, and Choudur Lakshminarayan. Nearest Neighbor Distributions for imbalanced classification. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–5, 2012. doi:10.1109/IJCNN.2012.6252718.

[124] Miroslav Kubat and Stan Matwin. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In *Proceedings of the 14th International Conference on Machine Learning (ICML 1997)*, pp. 179–186. Morgan Kaufmann, San Francisco, CA, 1997.

[125] Miroslav Kubat, Robert C. Holte, and Stan Matwin. Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*, 30(2):195–215, 1998. doi:10.1023/A:1007452223027.

[126] Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, pp. 2568–2573, 2011.

[127] Mathieu Latourrette. Toward an Explanatory Similarity Measure for Nearest-Neighbor Classification. In *Proceedings of the 11th European Conference on Machine Learning (ECML 2000)*, pp. 238–245. Springer, Heidelberg, 2000. doi:10.1007/3-540-45164-1_25.

[128] Jinyan Li, Guozhu Dong, Kotagiri Ramamohanarao, and Limsoon Wong. DeEPs: A New Instance-Based Lazy Discovery and Classification System. *Machine Learning*, 54(2):99–124, 2004. doi:10.1023/B:MACH.0000011804.08528.7d.

[129] Junnan Li, Qingsheng Zhu, Quanwang Wu, and Zhu Fan. A novel oversampling technique for class-imbalanced learning based on SMOTE and natural neighbors. *Information Sciences*, 565:438–455, 2021. doi:10.1016/j.ins.2021.03.041.

[130] Yuxuan Li and Xiuzhen Zhang. Improving k Nearest Neighbor with Exemplar Generalization for Imbalanced Classification. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2011)*, pp. 321–332. Springer, Heidelberg, 2011. doi:10.1007/978-3-642-20847-8_27.

[131] Moshe Lichman. UCI Machine Learning Repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[132] Charles X. Ling, Qiang Yang, Jianning Wang, and Shichao Zhang. Decision Trees with Minimal Costs. In *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*, pp. 69–77. ACM, New York, NY, 2004. doi:10.1145/1015330.1015369.

[133] Zachary C. Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. Optimal Thresholding of Classifiers to Maximize F1 Measure. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2014)*, pp. 225–239. Springer, Heidelberg, 2014. doi:10.1007/978-3-662-44851-9_15.

[134] Huan Liu and Hiroshi Motoda. *Computational Methods of Feature Selection*. Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC, Boca Raton, FL, 1st edition, 2007. doi:10.1201/9781584888796.

[135] Wei Liu and Sanjay Chawla. Class Confidence Weighted kNN Algorithms for Imbalanced Data Sets. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2011)*, pp. 345–356. Springer, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-20847-8_29.

[136] Wei Liu, Sanjay Chawla, David A. Cieslak, and Nitesh V. Chawla. A Robust Decision Tree Algorithm for Imbalanced Data Sets. In *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM)*, pp. 766–777, 2010. doi:10.1137/1.9781611972801.67.

[137] Yang Liu, Boqin Feng, and Guohua Bai. Compact Rule Learner on Weighted Fuzzy Approximation Spaces for Class Imbalanced and Hybrid Data. In *Rough Sets and Current Trends in Computing (RSCTC 2008)*, pp. 262–271. Springer, Heidelberg, 2008. doi:10.1007/978-3-540-88425-5_27.

[138] Victoria López, Alberto Fernández, Jose G. Moreno-Torres, and Francisco Herrera. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics. *Expert Systems with Applications*, 39(7):6585–6608, 2012. doi:10.1016/j.eswa.2011.12.043.

[139] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013. doi:10.1016/j.ins.2013.07.007.

[140] Victoria López, Alberto Fernández, and Francisco Herrera. On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences*, 257:1–13, 2014. doi:10.1016/j.ins.2013.09.038.

[141] Tomasz Maciejewski and Jerzy Stefanowski. Local neighbourhood extension of SMOTE for mining imbalanced data. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pp. 104–111, 2011. doi:10.1109/CIDM.2011.5949434.

[142] Larry Manevitz and Malik Yousef. One-class document classification via Neural Networks. *Neurocomputing*, 70(7):1466–1481, 2007. doi:10.1016/j.neucom.2006.05.013.

[143] Larry M. Manevitz and Malik Yousef. One-Class SVMs for Document Classification. *Journal of Machine Learning Research*, 2:139–154, 2002.

[144] Antonio Maratea, Alfredo Petrosino, and Mario Manzo. Adjusted F-measure and kernel scaling for imbalanced data learning. *Information Sciences*, 257:331–341, 2014. doi:10.1016/j.ins.2013.04.016.

[145] Michał Marcinkowski. *Construction of Classifiers for Imbalanced Data in Medical Applications (in Polish)*. Master's thesis, Politechnika Poznańska, Poznań, 2005.

[146] Ryszard S. Michalski. A Theory and Methodology of Inductive Learning. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach*, pp. 83–134. Springer, Heidelberg, 1983. doi:10.1007/978-3-662-12405-5_4.

[147] Ryszard S. Michalski, Igor Mozetic, Jiarong Hong, and Nada Lavrac. The Multi-Purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains. In *Proceedings of the 5th AAAI National Conference on Artificial Intelligence*, pp. 1041–1045. AAAI Press, 1986.

[148] Claudia R. Milaré, Gustavo E. A. P. A. Batista, and André C. P. L. F. Carvalho. A hybrid approach to learn with imbalanced classes using evolutionary algorithms. *Logic Journal of the IGPL*, 19(2):293–303, 2011. doi:10.1093/jigpal/jzq027.

[149] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997.

[150] Dunja Mladenic and Marko Grobelnik. Feature Selection for Unbalanced Class Distribution and Naive Bayes. In *Proceedings of the 16th International Conference on Machine Learning (ICML 1999)*, pp. 258–267. Morgan Kaufmann, San Francisco, CA, 1999.

[151] Aldi A. Nababan, Opim S. Sitompul, and Tulus. Attribute Weighting Based K-Nearest Neighbor Using Gain Ratio. *Journal of Physics: Conference Series*, 1007(1):Article No. 012007, 1–6, 2018. doi:10.1088/1742-6596/1007/1/012007.

[152] Krystyna Napierała. *Improving Rule Classifiers For Imbalanced Data*. PhD thesis, Poznań University of Technology, Poznań, 2012.

[153] Krystyna Napierała and Jerzy Stefanowski. BRACID: a comprehensive approach to learning rules from imbalanced data. *Journal of Intelligent Information Systems*, 39(2):335–373, 2012. doi:10.1007/s10844-011-0193-0.

[154] Krystyna Napierała and Jerzy Stefanowski. Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, 46(3):563–597, 2016. doi:10.1007/s10844-015-0368-1.

[155] Krystyna Napierała, Jerzy Stefanowski, and Szymon Wilk. Learning from Imbalanced Data in Presence of Noisy and Borderline Examples. In *Rough Sets and Current Trends in Computing (RSCTC 2010)*, pp. 158–167. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-13529-3_18.

[156] Peter Björn Nemenyi. *Distribution-free Multiple Comparisons*. PhD thesis, Princeton University, Princeton, NJ, 1963.

[157] Canh Hao Nguyen and Tu Bao Ho. An Imbalanced Data Rule Learner. In *Knowledge Discovery in Databases: PKDD 2005*, pp. 617–624. Springer, Heidelberg, 2005. doi:10.1007/11564126_65.

[158] Hung Son Nguyen. Approximate Boolean Reasoning: Foundations and Applications in Data Mining. In James F. Peters and Andrzej Skowron (eds.), *Transactions on Rough Sets V*, pp. 334–506. Springer, Heidelberg, 2006. doi:10.1007/11847465_16.

[159] Hung Son Nguyen and Sinh Hoa Nguyen. Discretization Methods in Data Mining. In Polkowski and Skowron [166], pp. 451–482.

[160] Sinh Hoa Nguyen. Regularity Analysis and its Applications in Data Mining. In Polkowski et al. [167], pp. 289–378. doi:10.1007/978-3-7908-1840-6_7.

[161] Son H. Nguyen and Andrzej Skowron. Quantization Of Real Value Attributes – Rough Set and Boolean Reasoning Approach. In *Proceedings of the 2nd Joint Annual Conference on Information Sciences (JCIS 1995)*, pp. 34–37, 1995.

[162] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. Available at: `http://neuralnetworksanddeeplearning.com`.

[163] Shameem A. Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. Optimizing F-Measures by Cost-Sensitive Classification. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS 2014)*, pp. 2123–2131. MIT Press, Cambridge, MA, 2014.

[164] Zdzisław Pawlak and Andrzej Skowron. A Rough Set Approach to Decision Rules Generation. In *Proceedings of the Workshop W12: The Management of Uncertainty at the 13th International Joint Conference on Artificial Intelligence (IJCAI 1993)*, pp. 114–119. Morgan Kaufmann, Chambéry, 1993.

[165] Judea Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009. doi:10.1214/09-SS057.

[166] Lech Polkowski and Andrzej Skowron (eds.). *Rough Sets in Knowledge Discovery 1: Methodology and Applications*, volume 18 of *Studies in Fuzziness and Soft Computing*. Physica-Verlag, Heidelberg, 1998.

[167] Lech Polkowski, Shusaku Tsumoto, and Tsau Y. Lin (eds.). *Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems*, volume 56 of *Studies in Fuzziness and Soft Computing*. Physica-Verlag, Heidelberg, 2000. doi:10.1007/978-3-7908-1840-6.

[168] Ronaldo C. Prati, Gustavo E. A. P. A. Batista, and Maria Carolina Monard. Class Imbalances versus Class Overlapping: An Analysis of a Learning System Behavior. In *Advances in Artificial Intelligence (MICAI 2004)*, pp. 312–321. Springer, Heidelberg, 2004. doi:10.1007/978-3-540-24694-7_32.

[169] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[170] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, 2017. URL `https://www.R-project.org`.

[171] Troy Raeder, T. Ryan Hoens, and Nitesh V. Chawla. Consequences of Variability in Classifier Performance Estimates. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM)*, pp. 421–430, 2010. doi:10.1109/ICDM.2010.110.

[172] Troy Raeder, George Forman, and Nitesh V. Chawla. Learning from Imbalanced Data: Evaluation Matters. In Dawn E. Holmes and Lakhmi C. Jain (eds.), *Data Mining: Foundations and Intelligent Paradigms: Volume 1: Clustering, Association and Classification*, pp. 315–331. Springer, Heidelberg, 2012. doi:10.1007/978-3-642-23166-7_12.

[173] Bhagat Singh Raghuwanshi and Sanyam Shukla. SMOTE based class-specific extreme learning machine for imbalanced learning. *Knowledge-Based Systems*, 187:Article No. 104814, 1–17, 2020. doi:10.1016/j.knosys.2019.06.022.

[174] Bhavani Raskutti and Adam Kowalczyk. Extreme Re-Balancing for SVMs: A Case Study. *ACM SIGKDD Explorations Newsletter*, 6(1):60–69, 2004. doi:10.1145/1007730.1007739.

[175] Michael M. Richter and Rosina Weber. *Case-Based Reasoning*. Springer-Verlag, Heidelberg, 1st edition, 2013. doi:10.1007/978-3-642-40167-1.

[176] Patricia Riddle, Richard Segal, and Oren Etzioni. Representation Design and Brute-force Induction in a Boeing Manufacturing Domain. *Applied Artificial Intelligence*, 8(1):125–147, 1994. doi:10.1080/08839519408945435.

[177] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, Hoboken, NJ, 4th edition, 2021.

[178] Corneliu Rusu and Lacrimioara Grama. Recent developments in acoustical signal classification for monitoring. In *2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE)*, pp. 1–10, 2017. doi:10.1109/ISEEE.2017.8170705.

[179] José A. Sáez, Julián Luengo, Jerzy Stefanowski, and Francisco Herrera. SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Information Sciences*, 291:184–203, 2015. doi:10.1016/j.ins.2014.08.051.

[180] Indu Saini, Dilbag Singh, and Arun Khosla. QRS detection using K-Nearest Neighbor algorithm (KNN) and evaluation on standard ECG databases. *Journal of Advanced Research*, 4(4):331–344, 2013. doi:10.1016/j.jare.2012.05.007.

[181] Steven L. Salzberg. On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach. *Data Mining and Knowledge Discovery*, 1(3):317–328, 1997. doi:10.1023/A:1009752403260.

[182] Claude Sammut and Geoffrey I Webb (eds.). *Encyclopedia of Machine Learning and Data Mining*. Springer, USA, 2nd edition, 2017. doi:10.1007/978-1-4899-7687-1.

[183] Andrzej Skowron. Boolean reasoning for decision rules generation. In *Methodologies for Intelligent Systems (ISMIS 1993)*, pp. 295–305. Springer, Heidelberg, 1993. doi:10.1007/3-540-56804-2_28.

[184] Andrzej Skowron and Cecylia Rauszer. The Discernibility Matrices and Functions in Information Systems. In Roman Słowiński (ed.), *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, pp. 331–362. Springer, Dordrecht, 1992. doi:10.1007/978-94-015-7975-9_21.

[185] Andrzej Skowron and Arkadiusz Wojna. K Nearest Neighbor Classification with Local Induction of the Simple Value Difference Metric. In *Rough Sets and Current Trends in Computing (RSCTC 2004)*, pp. 229–234. Springer, Heidelberg, 2004. doi:10.1007/978-3-540-25929-9_27.

[186] Yang Song, Jian Huang, Ding Zhou, Hongyuan Zha, and C. Lee Giles. IKNN: Informative K-Nearest Neighbor Pattern Classification. In *Knowledge Discovery in Databases (PKDD 2007)*, pp. 248–264. Springer, Heidelberg, 2007. doi:10.1007/978-3-540-74976-9_25.

[187] Craig Stanfill and David Waltz. Toward Memory-Based Reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986. doi:10.1145/7902.7906.

[188] Jerzy Stefanowski. Rough set based rule induction techniques for classification problems. In *Proceedings of 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT 1998)*, volume 1, pp. 109–113. Verlag Mainz, Aachen, 1998.

[189] Jerzy Stefanowski. Algorithms of rule induction for knowledge discovery (in Polish). Habilitation Thesis, 2001.

[190] Jerzy Stefanowski. On Combined Classifiers, Rule Induction and Rough Sets. In James F. Peters, Andrzej Skowron, Ivo Düntsch, Jerzy Grzymała-Busse, Ewa Orłowska, and Lech Polkowski (eds.), *Transactions on Rough Sets VI: Commemorating the Life and Work of Zdzisław Pawlak, Part I*, pp. 329–350. Springer, Heidelberg, 2007. doi:10.1007/978-3-540-71200-8_18.

[191] Jerzy Stefanowski. Overlapping, Rare Examples and Class Decomposition in Learning Classifiers from Imbalanced Data. In Sheela Ramanna, Lakhmi C. Jain, and Robert J. Howlett (eds.), *Emerging Paradigms in Machine Learning*, pp. 277–306. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-28699-5_11.

[192] Jerzy Stefanowski and Szymon Wilk. Rough sets for handling imbalanced data: Combining filtering and rule-based classifiers. *Fundamenta Informaticae*, 72(1-3):379–391, 2006.

[193] Jerzy Stefanowski and Szymon Wilk. Extending Rule-Based Classifiers to Improve Recognition of Imbalanced Classes. In Zbigniew W. Ras and Agnieszka Dardzinska (eds.), *Advances in Data Management*, pp. 131–154. Springer, Heidelberg, 2009. doi:10.1007/978-3-642-02190-9_7.

[194] Chao-Ton Su and Yu-Hsiang Hsiao. An Evaluation of the Robustness of MTS for Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 19(10):1321–1332, 2007. doi:10.1109/TKDE.2007.190623.

[195] Yanmin Sun, Andrew K. C. Wong, and Mohamed S. Kamel. Classification of Imbalanced Data: A Review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23 (4):687–719, 2009. doi:10.1142/S0218001409007326.

[196] Muhammad Atif Tahir, Josef Kittler, Krystian Mikolajczyk, and Fei Yan. A Multiple Expert Approach to the Class Imbalance Problem Using Inverse Random under Sampling. In *Multiple Classifier Systems (MCS 2009)*, pp. 82–91. Springer, Heidelberg, 2009. doi:10.1007/978-3-642-02326-2_9.

[197] Deepika Tiwari. Handling Class Imbalance Problem Using Feature Selection. *International Journal of Advanced Research in Computer Science & Technology*, 2(2):516–520, 2014.

[198] Bogdan Trawiński, Magdalena Smętek, Zbigniew Telec, and Tadeusz Lasota. Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms. *International Journal of Applied Mathematics and Computer Science*, 22(4):867–881, 2012. doi:10.2478/v10006-012-0064-z.

[199] Leslie Valiant. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books, Inc., New York, NY, 2013.

[200] Leslie G. Valiant. Robust logics. *Artificial Intelligence*, 117(2):231–253, 2000. doi:10.1016/S0004-3702(00)00002-3.

[201] Peter van der Putten and Maarten van Someren. A Bias-Variance Analysis of a Real World Learning Problem: The CoIL Challenge 2000. *Machine Learning*, 57(1):177–195, 2004. doi:10.1023/B:MACH.0000035476.95130.99.

[202] Gitte Vanwinckelen and Hendrik Blockeel. On estimating model accuracy with repeated cross-validation. In *Proceedings of the 21st Belgian-Dutch Conference on Machine Learning (BeneLearn 2012)*, pp. 39–44. Ghent, 2012.

[203] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, New York, NY, 1st edition, 1998.

[204] Nele Verbiest, Enislay Ramentol, Chris Cornelis, and Francisco Herrera. Preprocessing noisy imbalanced datasets using SMOTE enhanced with fuzzy rough prototype selection. *Applied Soft Computing*, 22:511–517, 2014. doi:10.1016/j.asoc.2014.05.023.

[205] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS 2016)*, pp. 3637–3645. Curran Associates Inc., Red Hook, NY, 2016.

[206] Pattaramon Vuttipittayamongkol and Eyad Elyan. Neighbourhood-based undersampling approach for handling imbalanced and overlapped data. *Information Sciences*, 509:47–70, 2020. doi:10.1016/j.ins.2019.08.062.

[207] Shuo Wang, Leandro L. Minku, and Xin Yao. A Systematic Study of Online Class Imbalance Learning With Concept Drift. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4802–4821, 2018. doi:10.1109/TNNLS.2017.2771290.

[208] Ronald L. Wasserstein and Nicole A. Lazar. The ASA Statement on p-Values: Context, Process, and Purpose. *The American Statistician*, 70(2):129–133, 2016. doi:10.1080/00031305.2016.1154108.

[209] Gary M. Weiss. Mining with rarity: a unifying framework. *ACM SIGKDD Explorations Newsletter*, 6(1):7–19, 2004. doi:10.1145/1007730.1007734.

[210] Gary M. Weiss. The Impact of Small Disjuncts on Classifier Learning. In Robert Stahlbock, Sven F. Crone, and Stefan Lessmann (eds.), *Data Mining: Special Issue in Annals of Information Systems*, pp. 193–226. Springer, Boston, MA, 2010. doi:10.1007/978-1-4419-1280-0_9.

[211] Gary M. Weiss and Foster Provost. Learning When Training Data Are Costly: The Effect of Class Distribution on Tree Induction. *Journal of Artificial Intelligence Research*, 19(1): 315–354, 2003.

[212] Dietrich Wettschereck, David W. Aha, and Takao Mohri. A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. *Artificial Intelligence Review*, 11(1):273–314, 1997. doi:10.1023/A:1006593614256.

[213] D. Randall Wilson and Tony R. Martinez. Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research*, 6(1):1–34, 1997. doi:10.1613/jair.346.

[214] Dennis L. Wilson. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2(3):408–421, 1972. doi:10.1109/TSMC.1972.4309137.

[215] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*. Morgan Kaufmann, 4th edition, 2016. doi:10.1016/b978-0-12-804291-5.00024-6.

[216] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Cambridge, MA, 4th edition, 2017. doi:10.1016/C2015-0-02071-8.

[217] Arkadiusz Wojna. Center-based indexing for nearest neighbors search. In *3rd IEEE International Conference on Data Mining (ICDM 2003)*, pp. 681–684, 2003. doi:10.1109/ICDM.2003.1251007.

[218] Arkadiusz Wojna. Center-Based Indexing in Vector and Metric Spaces. *Fundamenta Informaticae*, 56(3):285–310, 2003.

[219] Arkadiusz Wojna. Analogy-Based Reasoning in Classifier Construction. In James F. Peters and Andrzej Skowron (eds.), *Transactions on Rough Sets IV*, pp. 277–374. Springer, Heidelberg, 2005. doi:10.1007/11574798_11.

[220] Arkadiusz Wojna. Combination of Metric-Based and Rule-Based Classification. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing (RSFDGrC 2005)*, pp. 501–511. Springer, Heidelberg, 2005. doi:10.1007/11548669_52.

[221] Arkadiusz Wojna and Rafał Latkowski. Rseslib 3: Library of Rough Set and Machine Learning Methods with Extensible Architecture. In James F. Peters and Andrzej Skowron (eds.), *Transactions on Rough Sets XXI*, pp. 301–323. Springer, Berlin, Heidelberg, 2019. doi:10.1007/978-3-662-58768-3_7.

[222] Arkadiusz Wojna, Rafał Latkowski, and Łukasz Kowalski. *RSESLIB: User Guide*, 2019. URL `http://rseslib.mimuw.edu.pl/rseslib.pdf`.

[223] David H. Wolpert. The Supervised Learning No-Free-Lunch Theorems. In Rajkumar Roy, Mario Klöppen, Seppo Ovaska, Takeshi Furuhashi, and Frank Hoffmann (eds.), *Soft Computing and Industry: Recent Applications*, pp. 25–42. Springer, London, 2002. doi:10.1007/978-1-4471-0123-9_3.

[224] Kevin S. Woods, Christopher C. Doss, Kevin W. Bower, Jefferey L. Solka, Carey E. Priebe, and W. Philip Kegelmeyer. Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(6):1417–1436, 1993. doi:10.1142/S0218001493000698.

[225] Jakub Wróblewski. Covering with Reducts – A Fast Algorithm for Rule Generation. In *Rough Sets and Current Trends in Computing (RSCTC 1998)*, pp. 402–407. Springer, Heidelberg, 1998. doi:10.1007/3-540-69115-4_55.

[226] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey McLachlan, Angus Ng, Bing Liu, Philip Yu, Zhi-Hua Zhou, Michael Steinbach, David Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008. doi:10.1007/s10115-007-0114-2.

[227] Wendong Xiao, Jie Zhang, Yanjiao Li, Sen Zhang, and Weidong Yang. Class-specific cost regulation extreme learning machine for imbalanced classification. *Neurocomputing*, 261:70–82, 2017. doi:10.1016/j.neucom.2016.09.120.

[228] Yilin Yan, Min Chen, Mei-Ling Shyu, and Shu-Ching Chen. Deep Learning for Imbalanced Multimedia Data Classification. In *IEEE International Symposium on Multimedia (ISM 2015)*, pp. 483–488, 2015. doi:10.1109/ISM.2015.126.

[229] Qiang Yang and Xindong Wu. 10 Challenging Problems in Data Mining Research. *International Journal of Information Technology & Decision Making*, 05(04):597–604, 2006. doi:10.1142/S0219622006002258.

[230] Tao Yang, Longbing Cao, and Chengqi Zhang. A Novel Prototype Reduction Method for the K-Nearest Neighbor Algorithm with K≥1. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2010)*, pp. 89–100. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-13672-6_10.

[231] Jaesub Yun, Jihyun Ha, and Jong-Seok Lee. Automatic Determination of Neighborhood Size in SMOTE. In *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication (IMCOM 2016)*. ACM, New York, NY, 2016. doi:10.1145/2857546.2857648.

[232] Lotfi A. Zadeh. From Computing with Numbers to Computing with Words – From Manipulation of Measurements to Manipulation of Perceptions. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 45(1):105–119, 1999. doi:10.1109/81.739259.

[233] Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. In *3rd IEEE International Conference on Data Mining (ICDM 2003)*, pp. 435–442, 2003. doi:10.1109/ICDM.2003.1250950.

[234] Luis E. Zárate, Bruno M. Nogueira, Tadeu R. A. Santos, and Mark A. J. Song. Techniques for Missing Value Recovering in Imbalanced Databases: Application in a Marketing Database with Massive Missing Data. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pp. 2658–2664, 2006. doi:10.1109/ICSMC.2006.385265.

[235] Cha Zhang and Yunqian Ma. *Ensemble Machine Learning: Methods and Applications*. Springer-Verlag, New York, NY, 1st edition, 2012.

[236] Jianping Zhang, Eric Bloedorn, Lowell Rosen, and Daniel Venese. Learning rules from highly unbalanced data sets. In *4th IEEE International Conference on Data Mining (ICDM 2004)*, pp. 571–574, 2004. doi:10.1109/ICDM.2004.10015.

[237] Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Ruili Wang. Efficient kNN Classification With Different Numbers of Nearest Neighbors. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1774–1785, 2018. doi:10.1109/TNNLS.2017.2673241.

[238] Yong Zhang and Dapeng Wang. A Cost-Sensitive Ensemble Method for Class-Imbalanced Datasets. *Abstract and Applied Analysis*, 2013:Article No. 196256, 1–6, 2013. doi:10.1155/2013/196256.

[239] Qingsheng Zhu, Ji Feng, and Jinlong Huang. Natural neighbor: A self-adaptive neighborhood method without parameter K. *Pattern Recognition Letters*, 80:30–36, 2016. doi:10.1016/j.patrec.2016.05.007.

[240] Konstantin Zuev. Statistical Inference. [online], 2018. Available at SSRN: https://ssrn.com/abstract=3125891 or http://dx.doi.org/10.2139/ssrn.3125891.

Index

Abbreviations

A – set of (conditional) attributes, 23
a – attribute (usually conditional attribute), 23
$Agr(\{\varrho_a\}_{a\in A})$ – aggregated pseudometric, 30
A_{num} – set of numerical attributes, 23
A_{sym} – set of symbolic attributes, 23
AUC- Area Under the ROC Curve, 56

BRACID – Bottom-up induction of Rules And Cases for Imbalanced Data, 50, 145

CHM– City And Hamming Metric, 31
CombRules – combined rules, 34
c-rule(tst, trn) – combined local decision rule, 65
c-rule – combined local decision rule, 65
CSVDM – City And Simplified Value Difference pseudoMetric, 30

d – decision attribute, 24
$decision_{kNN}(tst)$ – kNN classifier, 40
$decisionLocal_{MaxRules}(tst, k, \varrho)$ – classifier based on maximally general rules with the support counted locally, 72
$decision_{MaxRules}(tst)$ – classifier based on maximally general rules, 36
defF – *def* strategy using F-measure, 156, 173
defG – *def* strategy using G-mean measure, 156, 164
d_{maj} – majority class, 24
d_{min} – minority class, 24

`ENN` – filter, Edited Nearest Neighbour, 148

$GenRules, GenRules\left(\{(\varrho_a, c_a)\}_{a\in A_{sym}}\right)$ – general rules, 34
g-rule (tst, trn), *g-rule* $\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}\right)$ – generalised local decision rule, 68
g-rule – generalised local decision rule, 68

J48 – learning algorithm, 146

K_{def} – default set for K in RIONIDA, 125
k_{max} – maximal possible value of parameter k, 81, 122
kNN – k-nearest neighbours, 39, 40, 145

l_a – lower bound of values from V_a for numerical attribute a, 23
LAZY – simple lazy rule induction algorithm for symbolic attributes, 37
$LocalStrength(tst, v, k, \varrho)$ – local measure for conflict resolution, 72

maxF – *max* strategy using F-measure, 156, 173
maxG – *max* strategy using G-mean measure, 156, 164
$MaxRules, MaxRules(Rules, trnSet)$ – set of maximally general rules, 35
ML – Machine Learning, 9
MODLEM – learning algorithm, 146
MODLEM-C – learning algorithm for imbalanced data, 145

N – neighbourhood of an example, 40
$N(tst, trnSet, k, \varrho)$, $N(tst, k)$ – neighbourhood of the example tst, 39
`Null-filter` – trivial configuration of filters (no filter), 148

ONIDA – Optimal Neighbourhood for Imbalanced Data Algorithm, 101
ONN – Optimal Nearest Neighbour algorithm, 84
optF – *opt* strategy using F-measure, 156, 173

List of Symbols

$[[\ldots]]_{\mathbb{D}}$ semantics of elementary condition or premise of rule; denotes a subset of $\mathbf{X}$

$[[\ldots]]_{trnSet}$ semantics of elementary condition or premise of rule restricted to training set $trnSet$

$[b, e]$ or $(b, e]$ or $[b, e)$ or (b, e) description of elementary set for numerical attributes representing interval between points b and e

$[b, e]$ or $(b, e]$ or $[b, e)$ or (b, e) interval between points b and e

α significance level (for statistical tests)

$\bigcup$ union of family of sets

$\mathbf{E}R$ expected value of random variable R

$\mathbf{E}_{z\sim\mathcal{D}}R(z)$ expected value of random variable R, where sampling of z is according to the probability distribution $\mathcal{D}$

$\emptyset$ (description of) the empty set

$|\ldots|$ cardinality (size) of a set

$||\ldots||_{\mathbb{D}}$ semantics of description of elementary set; for attribute a it is a subset of V_a

$\lfloor v \rfloor$ the 'floor' under v, i.e. the greatest integer less than or equal to v

$\mathbb{D}$ (pseudometric) decision system

$\mathbb{N}$ set of Natural Numbers

$\mathbb{R}$ set of Real Numbers

$\Pr_{z\sim\mathcal{D}}(\cdot \mid \cdot)$ conditional probability

$\Pr_{z\sim\mathcal{D}}(Event(z))$ probability of the event $Event$, where sampling of z is according to the probability distribution $\mathcal{D}$

$\prod$ Carthesian product of family of sets

$\sum$ sum of multiple real numbers

$\inf_{p\in P} f(p)$ infimum of $f(p)$ over $p \in P$

$\sup_{p \in P} f(p)$ supremum of $f(p)$ over $p \in P$

$\arg\max_{u \in U} f(u)$ the point of the given (finite) domain U at which the value of function f is maximised (we assume that one such point exists; in the other case tie-breaking procedure is or should be specified)

$\arg\min_{u \in U} f(u)$ set of all points of the given domain U at which the values of function f are minimised (in the case when the set is a singleton set we don't distinguish between this set and its element)

ϱ pseudometric function, $\varrho : X \times X \to \mathbb{R}$

ϱ_a pseudometric function for attribute a, $\varrho_a : V_a \times V_a \to \mathbb{R}$

$\mathbf{X}$ space of objects (examples, cases), domain of learning

$\{\dots\}_{a \in X}$ set contatining elements indexed by elements of another set, e.g. $\{\varrho_a\}_{a \in A}$ – set of indexed pseudometrics

$\{v\}$ (description of) singleton set

A set of (conditional) attributes

a attribute (usually conditional attribute)

$a(x)$ value of a on object $x \in \mathbf{X}$

$a = *$ trivial condition, i.e. condition equivalent to $a \in V_a$

A_{num} set of numerical attributes

A_{sym} set of symbolic attributes

$B(n, p)$ binomial distribution with parameters n and p (number of trials and success probability for each trial, respectively)

$B(x, r)$ (description of) closed ball of radius r centred at x relative to a given pseudometric

$Class(d)$ objects with decision d

d decision attribute

d value of decision on example (only in Subsection 4.3.4)

$d(x)$ decision value on object $x \in \mathbf{X}$

d_{maj} majority class

d_{min} minority class

$F_{B(n,p)}(\cdot)$ cumulative distribution function of binomial distribution $B(n, p)$

$H(\cdot, \cdot)$ harmonic mean of its arguments

$I(\cdot)$ indicator function

$if\ t_1 \wedge t_2 \wedge \ldots \wedge t_m\ then\ d = v$ decision rule

K set of admissible values of the parameter k in RIONIDA

k neighbourhood size; parameter k (in particular, in RIONA and RIONIDA)

K_{def} default set for K in RIONIDA

k_{max} maximal possible value of k (used for learning phase)

l_a lower bound of values from V_a for numerical attribute a

m number of attributes in the training set

$max(a, b)$ maximum value from the two given numbers

$min(a, b)$ minimum value from the two given numbers

N neighbourhood of test example

n number of objects in the training set

$O(\cdot)$ order of time or space complexity

P set of admissible values of the parameter p in RIONIDA

p parameter p in RIONIDA, i.e. relative importance of minority class and majority class

$P(d = d_j \mid a = v)$ conditional decision probability given a value v of an attribute a

P_{def} default set for P in RIONIDA

$r_1 \Rightarrow r_2$ rule r_2 is more general than (or is implied by) a rule r_1 (see Definition 2.10)

S set of admissible values of the parameter s in RIONIDA

s parameter s in RIONIDA

S_{def} default set for S in RIONIDA

$T[i]$ i-th entry in table T

$t_a(r)$ condition t_i from Definition 2.6 of rule r corresponding to attribute a

$t_i \Rightarrow t$ condition t is more general than (or is implied by) a condition t_i (see Definition 2.10)

$t_i(r)$ i-th condition t_i from Definition 2.6 of rule r

trn training example

$trnSet$ training set

tst test example

u_a upper bound of values from V_a for numerical attribute a

V_a (description of) set of values of attribute a

V_d finite set of decisions

$X \times Y \times \ldots$ Carthesian product of two (or more) sets

$z \sim \mathcal{D}$ random sampling of z according to probability distribution $\mathcal{D}$